FREE I FAST I FOR ALL

ITU Journal

Future and evolving technologies

Special issue

AI and machine learning solutions in 5G and future networks

Volume 2 (2021), Issue 4

ITU Journal
Future and evolving technologies

Volume 2 (2021), Issue 4

AI and machine learning solutions in 5G and future networks

ITU Journal
Future and evolving technologies

FREE I FAST I FOR ALL

The ITU Journal on Future and Evolving Technologies (ITU J-FET) is an international journal providing complete coverage of all communications and networking paradigms, free of charge for both readers and authors.

The ITU Journal considers yet-to-be-published papers addressing fundamental and applied research. It shares new techniques and concepts, analyses and tutorials, and learnings from experiments and physical and simulated test beds. It also discusses the implications of the latest research results for policy and regulation, legal frameworks, and the economy and society. This publication builds bridges between disciplines, connects theory with application, and stimulates international dialogue. Its interdisciplinary approach reflects ITU's comprehensive field of interest and explores the convergence of ICT with other disciplines.

The ITU Journal welcomes submissions at any time, on any topic within its scope.

Publication rights

SUGGESTED CITATION: ITU Journal on Future and Evolving Technologies, Volume 2 (2021), Issue 4.

ADDITIONAL INFORMATION

Please visit the ITU J-FET website at
https://www.itu.int/en/journal/j-fet/Pages/default.aspx

Inquiries should be addressed to
Alessia Magliarditi at: journal@itu.int

Challenge Organizers' Editorial

Artificial Intelligence (AI) / Machine Learning (ML) is impacting every aspect of business and society. AI will also shape how communication networks, a lifeline of our society, will evolve.

Applying AI/ML in communication networks poses an entirely different set of challenges than in other domains like image recognition or natural language processing. Time scales in a communication network span many orders of magnitude; some parameters change on an annual basis (e.g. your subscription to a telecom provider), while others may vary on a millisecond timescale (e.g. resource block allocation in the radio access networks). In addition, network environments are dynamic and noisy. Limitations in computing resources in a network adds to these challenges. Thus, while telecom operators have seen early AI applications to predict customer churn, predict fraud, identify customers for promotions, the industry has been slower in applying AI in use cases related to different domains within networks like core networks, radio access networks and management domains.

ITU has been at the forefront to explore how to best apply AI/ML in future networks including 5G networks. To advance the use of AI/ML in the telco industry, the ITU AI/ML in 5G Challenge was born (https://aiforgood.itu.int/ai-ml-in-5g-challenge-2020/). It rallied like-minded students and professionals from around the globe to study the practical application of AI/ML in emerging and future networks. The first edition of the Challenge was conducted in 2020 with over 1300 students and professionals from 62 countries, competing for global recognition and a shared a prize fund totalling 33 000 CHF. Through the Challenge, ITU encourages and supports the growing community driving the integration of AI/ML in networks and at the same time enhances the community driving standardization work for AI/ML, creating new opportunities for industry and academia to influence the evolution of ITU standards. Tools, data resources and problem statements were contributed by industry and academia in Brazil, China, India, Ireland, Japan, Russia, Spain, Turkey and the United States. The Challenge offered participants an opportunity to showcase their talent, test their concepts on real data and real-world problems, and compete for global recognition. The solutions can be accessed in several repositories on the Challenge GitHub: https://github.com/ITU-AI-ML-in-5G-Challenge.

Many solutions submitted to the Challenge were innovative and, in some cases, improvements with respect to the baselines. To share the solutions with the larger community, ITU issued a call for papers for a special issue on AI and machine learning solutions in 5G and future networks of the ITU Journal on Future and Evolving Technologies (ITU J-FET). In this special issue, hosts (i.e., the originators of the problem statements) and participants of the ITU Challenge submitted their solutions and learnings for publication. This special issue is dedicated to exploration of Artificial Intelligence and Machine Learning in 5G and future networks as well as enabling technologies and tools in networks. After rigorous review by reviewers in conjunction with guest editors, 10 papers were accepted for publication.

The ability to automatically and rapidly detect network and device failures is an essential feature for network operators to provide reliable service in future networks and 5G. In the paper "*Analysis on route information failure in IP core networks by NFV-based test environment*," the authors propose a method that extract features from large-scale unstructured data to differentiate between normal and abnormal states. The proposed method reduces computation without degrading the performance and achieves a prediction accuracy of 94%.

Existing methods of network topology planning do not consider the increasing network traffic and uneven link capacity utilization, resulting in sub-optimal resource utilization and unnecessary investments in network construction. In this special issue, two papers "*Applying machine learning in network topology optimization*" and "*AI-based network topology optimization system*" consider the problem of topology optimization. The former proposes a solution by considering an ML pipeline in

ITU Y.3172 (https://www.itu.int/rec/T-REC-Y.3172/en, an ITU standard) and other mainstream algorithms to solve the problem, whereas the latter builds a Long Short-term Memory (LSTM) model for time series traffic forecasting, using NetworkX (a Python library) to dynamically optimize the network topology by edge deletion or addition based on traffic over nodes.

The paper "*Machine learning for performance prediction of channel bonding in next-generation IEEE 802.11 WLANS*" presents results gathered from the problem statement by Universitat Pompeu Fabra (UPF), whose primary goal was predicting the performance of next-generation Wireless Local Area Networks (WLANs) by applying Channel Bonding (CB) techniques. The paper presents an overview of ML models proposed by participants (including Artificial Neural Networks, Graph Neural Networks, Random Forest regression, and gradient boosting) and analyze their performance on an open dataset generated using the IEEE 802.11ax-oriented Komondor network simulator. The accuracy achieved by the proposed methods demonstrates the suitability of ML for predicting the performance of WLANs.

Recent advancements in Deep Learning (DL) have revolutionized the way we can efficiently tackle complex optimization problems. However, existing DL-based solutions are often considered as black boxes with high inner complexity. In this context, explainability techniques have recently emerged to unveil why DL models make each decision. The paper "*NetXplain: Real-time explainability of graph neural networks applied to networking*" focuses on the explainability of Graph Neural Networks (GNN) applied to networking. GNNs are a novel DL family with unique properties to generalize over graphs. As a result, they have shown unprecedented performance to solve complex network optimization problems. NetXplain is a novel real-time explainability solution that uses a GNN to interpret the output produced by another GNN. In evaluation, the proposed explainability method is applied to RouteNet, a GNN model that predicts end-to-end QoS metrics in networks.

In the paper "*Graph-neural-network-based delay estimation for communication networks with heterogeneous scheduling policies*," the authors propose a solution that supports multiple scheduling policies (Strict Priority, Deficit Round Robin, Weighted Fair Queuing) and handles mixed scheduling policies in a single communication network as opposed to RouteNet which is based on simplified assumptions (such as the restriction to a single scheduling policy). The solution proposed by the authors achieved a mean absolute percentage error under 1% on the evaluation data set from the Challenge. This takes neural-network-based delay estimation one step closer to practical use.

The paper titled "*Site-specific millimeter-wave compressive channel estimation algorithms with hybrid MIMO architectures*" presents and compares three novel model-cum-data driven channel estimation procedures in a millimeter-wave multi-input multi-output (MIMO) orthogonal frequency division multiplexing (OFDM) wireless communication system. The techniques are adapted from a wide range of signal processing methods, such as detection and estimation theories, compressed sensing, and Bayesian inference, to learn the unknown virtual beamspace domain dictionary, as well as the delay-and-beamspace sparse channel. The model-based algorithms were trained with a site-specific training dataset generated using a realistic ray tracing-based wireless channel simulation tool. Through benchmarking, model-based approaches combined with data-driven customization unanimously outperform the state-of-the-art techniques by a large margin.

Beamforming is an essential technology in the 5G massive multiple-input-multiple-output (MMIMO) communications, which are subject to many impairments due to the nature of the wireless transmission channel. The inter-cell interference (ICI) is one of the main obstacles faced by 5G communications due to frequency-reuse technologies. However, finding the optimal beamforming parameter to minimize the ICI requires infeasible prior network or channel information. The paper "*A dynamic Q-learning beamforming method for inter-cell interference mitigation in 5G massive MIMO networks*" proposes a dynamic Q-learning beamforming method for ICI mitigation in the 5G downlink that does not require prior network or channel knowledge. Comparing with a traditional beamforming method and other industrial Reinforcement Learning (RL) methods, the proposed method has lower computational

complexity and better convergence efficiency. Performance analysis shows the quality-of-service improvement in terms of signal-to-interference-plus-noise-ratio (SINR) and the robustness towards different environments.

The paper "*Enhanced shared experiences in heterogeneous network with generative AI*" considers an environment where the participants can interact with each other through video conferencing by only sending the audio in the network. The authors propose a multi-modal adaptive normalization-based architecture (MAN) to synthesize a talking person video of arbitrary length using as input an audio signal and a single image of a person. The architecture uses the multi-modal adaptive normalization, keypoint heatmap predictor, optical flow predictor and class activation map-based layers to learn movements of expressive facial components and hence generates a highly expressive talking-head video of the given person.

Digital representations of the real world are being used in many applications such as augmented reality. 6G systems will not only support use cases that rely on virtual worlds but also benefit from the rich contextual information to improve performance and reduce communication overhead. The paper "*Simulation of machine learning-based 6G systems in virtual worlds*" focuses on the simulation of 6G systems that rely on a 3-D representation of the environment, as captured by cameras and other sensors. New strategies for obtaining paired MIMO channels and multimodal data are presented and tradeoffs between speed and accuracy when generating channels via ray-tracing are discussed.

This special issue was made possible due to the tireless and selfless efforts by the Guest Editors. The leading Guest Editor – Chih-Lin I, China Mobile Research Institute, China – as well as the Guest Editors – Akihiro Nakao, University of Tokyo, Japan; Aldebaro Klautau, The Federal University of Pará (UFPA), Brazil; Nuria González Prelcic, North Carolina State University, USA; and Albert Cabellos-Aparicio, Technical University of Catalonia, Spain – worked together as a team to guide the authors. The insights, expert comments and recommendations of the well experienced Guest editors were invaluable for bringing out the innovations behind the Challenge in the form of this journal. We also thank the numerous reviewers who worked hard to make sure that we have quality manuscripts for this special issue. Furthermore, we would like to thank the authors who not only submitted solutions to the Challenge but also took the trouble to document them and share them to our readers. Last but not least we are grateful to the Editor-in-Chief of the ITU Journal, Ian Akyildiz, for his enthusiasm and guidance.

The second edition of the ITU AI/ML in 5G Challenge is already underway in 2021. This provides an opportunity for partners, hosts and participants to collaborate on new problem statements, datasets and solutions. The call for papers resulting from the second edition of the Challenge provides a further opportunity for collaboration and learning for our hosts and participants.

We invite you to enjoy reading the current special issue and to join us on our journey of the next one.

Vishnu Ram

Independent Researcher

Thomas Basikolo

ITU

Reinhard Scholl

ITU

EDITORIAL BOARD

TABLE OF CONTENTS

Page

LIST OF ABSTRACTS

Graph-neural-network-based delay estimation for communication networks with heterogeneous scheduling policies

Pages 1–8

Martin Happ, Matthias Herlich, Christian Maier, Jia Lei Du, Peter Dorfinger

Modeling communication networks to predict performance such as delay and jitter is important for evaluating and optimizing them. In recent years, neural networks have been used to do this, which may have advantages over existing models, for example from queueing theory. One of these neural networks is RouteNet, which is based on graph neural networks. However, it is based on simplified assumptions. One key simplification is the restriction to a single scheduling policy, which describes how packets of different flows are prioritized for transmission. In this paper we propose a solution that supports multiple scheduling policies (Strict Priority, Deficit Round Robin, Weighted Fair Queueing) and can handle mixed scheduling policies in a single communication network. Our solution is based on the RouteNet architecture as part of the "Graph Neural Network Challenge". We achieved a mean absolute percentage error under 1% with our extended model on the evaluation data set from the challenge. This takes neural-network-based delay estimation one step closer to practical use.

View Article

Site-specific millimeter-wave compressive channel estimation algorithms with hybrid MIMO architectures

Pages 9–26

Sai Subramanyam Thoota, Dolores Garcia Marti, Özlem Tugfe Demir, Rakesh Mundlamuri, Joan Palacios, Cenk M. Yetis, Christo Kurisummoottil Thomas, Sameera H. Bharadwaja, Emil Björnson, Pontus Giselsson, Marios Kountouris, Chandra R. Murthy, Nuria González-Prelcic, Joerg Widmer

In this paper, we present and compare three novel model-cum-data-driven channel estimation procedures in a millimeter-wave Multi-Input Multi-Output (MIMO) Orthogonal Frequency Division Multiplexing (OFDM) wireless communication system. The transceivers employ a hybrid analog-digital architecture. We adapt techniques from a wide range of signal processing methods, such as detection and estimation theories, compressed sensing, and Bayesian inference, to learn the unknown virtual beamspace domain dictionary, as well as the delay-and-beamspace sparse channel. We train the model-based algorithms with a site-specific training dataset generated using a realistic ray tracing-based wireless channel simulation tool. We assess the performance of the proposed channel estimation algorithms with the same site's test data. We benchmark the performance of our novel procedures in terms of normalized mean squared error against an existing fast greedy method and empirically show that model-based approaches combined with data-driven customization unanimously outperform the state-of-the-art techniques by a large margin. The proposed algorithms were selected as the top three solutions in the "ML5G-PHY Channel Estimation Global Challenge 2020" organized by the International Telecommunication Union.

View Article

Enhanced shared experiences in heterogeneous network with generative AI

Pages 27–46

Neeraj Kumar, Ankur Narang, Brejesh Lall, Nitish Kumar Singh

COVID-19 has made the immersive experiences such as video conferencing, virtual reality/augmented reality, the most important modes of exchanging information. Despite much advancement in the network bandwidth and codec techniques, the current system still suffers from glitches, lags and poor video quality, especially under unreliable network conditions. In this paper, we propose the method of a video streaming pipeline to provide better video quality under erratic network conditions. We propose an environment where the participants can interact with each other through video conferencing by only sending the audio in the network. We propose a Multimodal Adaptive Normalization (MAN)-based architecture to synthesize a talking person video of arbitrary length using as input: an audio signal and a single image of a person. The architecture uses multimodal adaptive normalization, keypoint heatmap predictor, optical flow predictor and class activation map-based layers to learn movements of expressive facial components and hence generates a highly expressive talking-head video of the given person. We demonstrate the effectiveness of proposed streaming that dynamically controls the Quality of Experience (QoE) as per the requirements.

View Article

A dynamic Q-learning beamforming method for inter-cell interference mitigation in 5G massive MIMO networks

Pages 47–55

Aidong Yang, Xinlang Yue, Mohan Wu, Ye Ouyang

Beamforming is an essential technology in 5G Massive Multiple-Input Multiple-Output (MMIMO) communications, which are subject to many impairments due to the nature of wireless transmission channel. The Inter-Cell Interference (ICI) is one of the main obstacles faced by 5G communications due to frequency-reuse technologies. However, finding the optimal beamforming parameter to minimize the ICI requires infeasible prior network or channel information. In this paper, we propose a dynamic Q-learning beamforming method for ICI mitigation in the 5G downlink that does not require prior network or channel knowledge. Compared with a traditional beamforming method and other industrial Reinforcement Learning (RL) methods, the proposed method has lower computational complexity and better convergence efficiency. Performance analysis shows the quality of service improvement in terms of Signal-to-Interference-plus-Noise-Ratio (SINR) and the robustness towards different environments.

View Article

NetXplain: Real-time explainability of graph neural networks applied to networking

Pages 57–66

David Pujol-Perich, José Suárez-Varela, Shihan Xiao, Bo Wu, Albert Cabellos-Aparicio, Pere Barlet-Ros

Recent advancements in Deep Learning (DL) have revolutionized the way we can efficiently tackle complex optimization problems. However, existing DL-based solutions are often considered as black boxes with high inner complexity. As a result, there is still certain skepticism among the networking industry about their practical viability to operate data networks. In this context, explainability techniques have recently emerged to unveil why DL models make each decision. This paper focuses on the explainability of Graph Neural Networks (GNNs) applied to networking. GNNs are a novel DL family with unique properties to generalize over graphs. As a result, they have shown unprecedented performance to solve complex network optimization problems. This paper presents NetXplain, a novel real-time explainability solution that uses a GNN to interpret the output produced by another GNN. In the evaluation, we apply the proposed explainability method to RouteNet, a GNN model that predicts end-to-end QoS metrics in networks. We show that NetXplain operates more than 3 orders of magnitude faster than state-of-the-art explainability solutions when applied to networks up to 24 nodes, which makes it compatible with real-time applications; while demonstrating strong capabilities to generalize to network scenarios not seen during training.

View Article

Machine learning for performance prediction of channel bonding in next-generation IEEE 802.11 WLANS

Pages 67–79

Francesc Wilhelmi, David Góez, Paola Soto, Ramon Vallés, Mohammad Alfaifi, Abdulrahman Algunayah, Jorge Martín-Pérez, Luigi Girletti, Rajasekar Mohan, K Venkat Ramnan, Boris Bellalta

With the advent of Artificial Intelligence (AI)-empowered communications, industry, academia, and standardization organizations are progressing on the definition of mechanisms and procedures to address the increasing complexity of future 5G and beyond communications. In this context, the International Telecommunication Union (ITU) organized the First AI for 5G Challenge to bring industry and academia together to introduce and solve representative problems related to the application of Machine Learning (ML) to networks. In this paper, we present the results gathered from Problem Statement 13 (PS-013), organized by Universitat Pompeu Fabra (UPF), whose primary goal was predicting the performance of next-generation Wireless Local Area Networks (WLANs) applying Channel Bonding (CB) techniques. In particular, we provide an overview of the ML models proposed by participants (including artificial neural networks, graph neural networks, random forest regression, and gradient boosting) and analyze their performance on an open data set generated using the IEEE 802.11ax-oriented Komondor network simulator. The accuracy achieved by the proposed methods demonstrates the suitability of ML for predicting the performance of WLANs. Moreover, we discuss the importance of abstracting WLAN interactions to achieve better results, and we argue that there is certainly room for improvement in throughput prediction through ML.

View Article

AI-based network topology optimization system

Pages 81–90

Han Zengfu, Kong Jiankun, Wang Zhiguo, Zhang Yiwei, Liu Ke, Pan Liang, Li Sicong, Wu Desheng

Existing network topology planning does not fully consider the increasing network traffic and problem of uneven link capacity utilization, resulting in lower resource utilization and unnecessary investments in network construction. The AI-based network topology optimization system introduced in this paper builds a Long Short-Term Memory (LSTM) model for time series traffic forecasting, which uses NetworkX, a Python library, for graph analysis, dynamically optimizes the network topology by edge deletion or addition based on traffic over nodes, and ensures network load balancing when node traffic increases, mainly introducing the LSTM forecasting model building process, parameter optimization strategy, and network topology optimization in some detail. As it effectively enhances resource utilization, this system is vital to the optimization of complex network topology. The end of this paper looks forward to the future development of artificial intelligence, and suggests the possibility of how to cooperate with operator networks and how to establish cross-border ecological development.

View Article

Applying machine learning in network topology optimization

Pages 91–99

Zhouwei Gang, Qianyin Rao, Lin Guo, Lin Xi, Zezhong Feng, Qian Deng

Nowadays, telecommunications have become an indispensable part of our life, 5G technology brings better network speeds, helps the AR and VR industry, and connects everything. It will deeply change our society. Transmission is the vessel of telecommunications. While the vessel is not so healthy, some of them are overloaded, meanwhile, others still have lots of capacity. It not only affects the customer experience, but also affects the development of communication services because of a resources problem. A transmission network is composed of transmission nodes and links. So that the possible topology numbers equal to node number multiplied by number of links means it is impossible for humans to optimize. We use Al instead of humans for topology optimization. The AI optimization solution uses an ITU Machine Learning (ML) standard, Breadth-First Search (BFS) greedy algorithm and other mainstream algorithms to solve the problem. It saves a lot of money and human resources, and also hugely improves traffic absorption capacity. The author comes from the team named "No Boundaries". The team attend ITU AI/ML in 5G Challenge and won the Gold champions (1st place).

View Article

Analysis on route information failure in IP core networks by NFV-based test environment

Xia Fei, Aerman Tuerxun, Jiaxing Lu, Ping Du, Akihiro Nakao

Stable and high-quality Internet connectivity is mandatory for 5G mobile networks. However, the pandemic of COVID-19 has forced global and large-scale staying at home and telecommuting in many countries. The increasing traffic has induced more pressure on networks, devices and cloud data centers. It becomes an essential task for network opera-tors to enable their ability to automatically and rapidly detect network and device failures. We propose a highly practical method based on highly practical technology. Our method has a high generalization ability that can efficiently extract features from large-scale unstructured data and ensure high accuracy prediction. First, 997 useful features are extracted from 28GB-per-day network logs. Then, a differential approach is employed to preprocess the extracted features so as to highlight the differences between normal and abnormal states. Third, those features are refined based on the feature importance we calculated. According to our experiment, the proposed feature extraction and refinement method can reduce computation without degrading the performance. Among the five types of failures, we achieve a 100% recall rate in four types and the rest can also reach 71%. Overall, the total average prediction accuracy of the proposed method is 94%.

View Article

Simulation of machine learning-based 6G systems in virtual worlds

Ailton Oliveira, Felipe Bastos, Isabela Trindade, Walter Frazão, Arthur Nascimento, Diego Gomes, Francisco Müller, Aldebaro Klautau

Digital representations of the real world are being used in many applications, such as augmented reality. 6G systems will not only support use cases that rely on virtual worlds but also benefit from their rich contextual information to improve performance and reduce communication overhead. This paper focuses on the simulation of 6G systems that rely on a 3D representation of the environment, as captured by cameras and other sensors. We present new strategies for obtaining paired MIMO channels and multimodal data. We also discuss trade-offs between speed and accuracy when generating channels via ray tracing. We finally provide beam selection simulation results to assess the proposed methodology.

View Article

GRAPH-NEURAL-NETWORK-BASED DELAY ESTIMATION FOR COMMUNICATION NETWORKS WITH HETEROGENEOUS SCHEDULING POLICIES

Martin Happ[1,2], Matthias Herlich[1], Christian Maier[1], Jia Lei Du[1], Peter Dorfinger[1]
[1]Intelligent Connectivity, Salzburg Research, Austria, [2]IDA LAB, University of Salzburg, Austria

NOTE: Corresponding authors: Martin Happ, martin.happ@sbg.ac.at; Peter Dorfinger, peter.dorfinger@salzburgresearch.at

Abstract – Modeling communication networks to predict performance such as delay and jitter is important for evaluating and optimizing them. In recent years, neural networks have been used to do this, which may have advantages over existing models, for example from queueing theory. One of these neural networks is RouteNet, which is based on graph neural networks. However, it is based on simplified assumptions. One key simplification is the restriction to a single scheduling policy, which describes how packets of different flows are prioritized for transmission. In this paper we propose a solution that supports multiple scheduling policies (Strict Priority, Deficit Round Robin, Weighted Fair Queueing) and can handle mixed scheduling policies in a single communication network. Our solution is based on the RouteNet architecture as part of the "Graph Neural Network Challenge". We achieved a mean absolute percentage error under 1% with our extended model on the evaluation data set from the challenge. This takes neural-network-based delay estimation one step closer to practical use.

Keywords – Communication networks, delay estimation, graph neural networks, scheduling

1. INTRODUCTION

There has been an increasing use of machine learning techniques for various kinds of problems in recent years. Due to the variety of problems, many new machine learning algorithms have been developed. In particular for data that can be described by graphs, there has been an important new development known as "Graph Neural Networks" [1]. Examples of such data are chemical elements or communication networks. We focus on the latter in this paper. In this context, a communication network can be characterized by nodes and edges, where the edges represent the links between nodes. Additionally, there are properties associated with each node and each edge. This is the basic setting of Graph Neural Networks (GNNs). GNNs use the so-called "Message Passing" algorithm [2] and can express the notion of nodes and edges. However, for communication networks it is also important to consider paths (and network flows) along several consecutive links. RouteNet [3, 4] is an implementation of this idea that allows expressing paths. The RouteNet architecture consists mainly of two gated recurrent neural networks that are responsible for calculating path and link properties, respectively.

The RouteNet architecture can be used to predict per-flow performance metrics such as average delay and jitter. This can be useful for assessing networks with respect to different loads without needing to test them in reality. Hence, it is possible to determine if a network can handle a certain load with respect to a performance metric such as average delay. An alternative to such a prediction with neural networks is a simulation, using simulators such as OMNeT++ [5]. However, such simulations may be time-consuming. If the communication network itself or any settings are changed, the simulation has to be repeated. Thus, it becomes especially time-consuming when simulating the impact of a series of parameter changes. In contrast, the time-consuming training of neural networks has to happen only once in general. Hence, prediction with neural networks usually provides a faster way to estimate the performance of networks.

2. RELATED WORK

There are classical (i.e. non-machine learning) methods to predict delays in communication networks, like queuing theory [6], network calculus [7] and simulation-based approaches [5].

Boutaba et al. [8] provide a general overview on the application of machine learning to communication technologies and network measurements. The approaches in particular differ in whether the data used for learning comes from network simulators (e.g. from OMNeT++ as in our case) or from actual measurements. In addition, they can be divided into supervised, unsupervised and reinforcement learning. The approach considered in this paper is an example of supervised learning.

Mestres et al. [9] investigate modeling and prediction of delays in communication networks with feed-forward neural networks. They predict the latency based on the traffic configuration. In contrast to the RouteNet architecture, a neural network has to be trained for each specific communication network. Graph neural networks and message passing were first introduced by Scarselli et

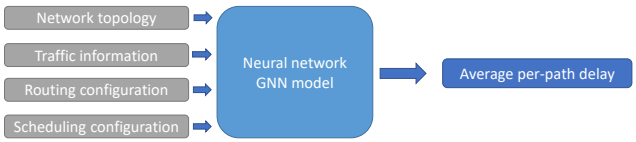


Fig. 1 – Problem setting for the challenge

al. [1] and Gilmer et al. [10]. These concepts were applied to the domain of communication technologies by Geyer and Carle [11], where the authors use a GNN for automatic network protocol design.

A different approach to deal with heterogeneous scheduling policies was recently proposed by Ferriol et al. [12]. They provide a GNN architecture with states for links, paths and queues. This model reaches a mean relative error of 3.88% in the German Backbone Network (GBN) topology (which was not used for the training process).

3. SETTING

The solution proposed in this paper was developed within the GNN Challenge [13] provided by the "Barcelona Neural Networking Center" from the Universitat Politècnica de Catalunya. This challenge was organized as part of the "ITU Artificial Intelligence/Machine Learning in 5G Challenge".

The goal of the GNN challenge was to predict the average per flow delay in a communication network. In other words, it is of interest to estimate the average time it takes for a packet to travel from its source node to its destination node. Additionally, the network topology may be different from that used in the training data. Thus, the neural network should not be adapted to only one topology but work for general topologies. For this, three different network topologies have been provided. For training, the NSFNET topology with 14 nodes [14] and GEANT2 topology with 24 nodes [15] were used. For validation, GBN [16] with 17 nodes was used. The data set that was used for the evaluation of the challenge consisted of 19 nodes. Other information has not been published by the challenge organizers. Thus, the proposed solution for this challenge must work even for such unknown communication networks where no details are known beforehand.

The data set consisted of different node and link information, as well as different settings used in the OMNeT++ simulator. A crucial simplification for all data sets is that there exists only one flow for each path. And for each flow, a Type of Service (ToS) is randomly assigned. That means all packets of a path have the same ToS. This is an important property of the data set and we will utilize it in Section 4.3. The number of generated packets per time unit follow a Poisson distribution, and the inter-packet arrival times follow an exponential distribution. A two-valued distribution is used to model packet size. The maximum bit rate is chosen randomly between 400 and 2000 bits per time unit. For more details on the simulated data, we refer to the challenge documentation [13].

In communication networks, scheduling policies describe in which order packets are transmitted. A simple and straightforward algorithm is FIFO, where the packets are transmitted in the order in which they are received [17]. The original RouteNet was developed for networks that use only a single scheduling policy. However, this implementation does not work well with other scheduling algorithms and networks with heterogeneous scheduling can have large a impact on the behavior of networks and thus on the delays. For this challenge specifically, three different scheduling policies, namely Weighted-Fair-Queuing (WFQ), Strict Priority (SP), and Deficit Round Robin (DRR) are being used. For WFQ and DRR, there are three ToS classes. The networks are in general not homogeneous with respect to scheduling policies, in fact there are data sets where all policies are present in a single communication network.

3.1 RouteNet

RouteNet uses Graph Neural Networks and the so-called message passing [2] for predicting average per-path delays in communication networks. There are two important elements in this architecture, links and paths; where each path consists of at least one link. Note that we use the term "capacity" to refer to the tight upper bound on the transmission rate of a link and the term "data rate" to refer to the desired transmission rate of a traffic flow on a network path. Each link is associated with specific information, such as link capacity. We will refer to this simply as link state information, which is represented as a vector. The same holds true for paths, which we will call analogously path state information. The RouteNet version provided for the challenge [18] uses at initialization only link capacity (bits/time unit) for the link state information h_l and the average data rate (bits/time unit) of a single flow for the path state information h_p. The data rate of the flow can be encoded as part of the path state information as in RouteNet it is assumed that there is at most one flow per path. The dimension of link and path information are both set to 32 in this RouteNet version, that is at the beginning only one component of the link and path state contains meaningful information, all other components are filled with zeros. Therefore, those two vectors can be written as

$$h_l = [x, 0, \dots, 0]' \in \mathbb{R}^{32} \text{ and}$$
$$h_p = [z, 0, \dots, 0]' \in \mathbb{R}^{32},$$

where x denotes the link capacity and z the average desired data rate on that path.

RouteNet utilizes two recurrent neural networks G_p and G_l. The neural network G_p calculates the new path state information based on the previous path information and link information. The result of this neural network is then used for G_l to calculate new link state information with the previous link state information. See Algorithm 1 for the pseudo-code. As G_p is a recurrent neural network, it returns the hidden path state after each link of a path.

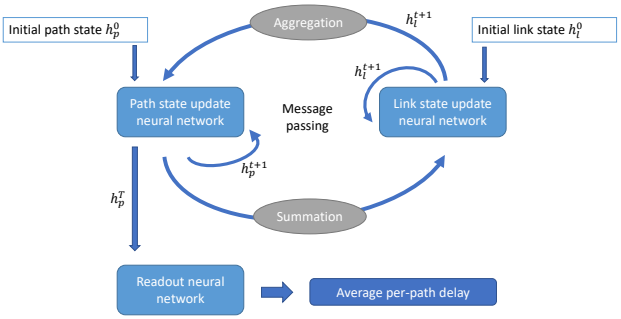


Fig. 2 – Schematic representation of RouteNet

All these states are combined to a sequence of path states for each path. Therefore, the output from G_p is aggregated for each link with a function that is denoted by f. In RouteNet, this f is equal to a summation. The function g reduces the output that is returned by G_p to the last state only, which is considered to be the new path state information. This algorithm or message passing between these two neural networks is repeated $T = 8$ times. The number of repetitions should be of the order of the average shortest path length [2]. The final path information is then an approximation of the fixed point of this message passing procedure. It is then used to predict the average delay with an additional neural network. Figure 2 gives a simplified overview of this message passing. Rusek et al. [4] give more details about the RouteNet architecture.

Data: path state h_p and link state vector h_l
Result: predicted per-path delay $\hat{y}_p$
for *t = 0 to T* **do**
  $H_p^{t+1} = G_p(h_p^t, h_l^t)$
  $h_l^{t+1} = G_l(f(H_p^{t+1}), h_l^t)$
  $h_p^{t+1} = g(H_p^{t+1})$
end
$\hat{y}_p = R(h_p^T)$;

Algorithm 1: RouteNet architecture

4. PROPOSED SOLUTION

Our proposed solution is a modification of RouteNet [3], which is based on message passing and graph neural networks. Instead of just providing the final architecture, we give an overview of all changes we applied to the original RouteNet model and provide intermediate results for the delay predictions. That way, it is possible to see and evaluate the impact that different changes had on the results. All variants have been repeated 5 times to also assess the stability and variability of each model. Note that this number of 5 replications is arbitrary and no sample size calculation was done to compare different variants with each other given a pre-specified power for the statistical analysis. We use 600 000 training steps for each run and an exponential decay after every 60 000 steps. That means the learning rate of 0.001 is multiplied by the factor 0.6 after 60 000 training steps. Regularization is the same as in the RouteNet implementation provided for the challenge [18], that is the L^2 regularization is set to 0.1 and 0.01 for all neural networks in the first hidden layer and second hidden layer of the readout neural network. In the following we illustrate the impact of all changes on the mean absolute percentage error.

4.1 Baseline

The task was to minimize the mean absolute percentage error of per-path delays. Hence, we decided to change the loss function in the original implementation from Mean Squared Error (MSE) to Mean Absolute Percentage Error (MAPE) to use the same metric for training and evaluation.
We compare this first change with the baseline code where the optimization is done with respect to the mean squared error. The results are displayed in Table 1 as Step 0 and Step 1. It shows that without any modifications the model does not perform well as the average error over 5 runs is over 200%. This is not surprising as the original RouteNet model was developed for networks with a different scheduling policy. Using the mean absolute percentage error as the target function improved the model significantly. The grand mean of all results was about 46% (with a 95% Confidence Interval (CI) of [26.5%, 66.29%]). This improvement was expected as the results were evaluated by the mean absolute percentage error and the training was done with the same target function.

4.2 Normalization

For neural networks, it is common and advised to standardize the input variables [19, 20]. Therefore, all variables were shifted into $[0, 1]$ such that they are on the same scale. No centering was applied. This modification significantly improved the results given in Table 1. The grand mean is about 23% (95% CI [23.7%, 23.74%]). It shows again, what is already known in the literature, that normalizing or standardizing input variables is crucial and should be done. Not only to improve prediction but also to improve stability of training the model, which is reflected in a small standard deviation of those 5 runs.

4.3 Adding variables

In Step 3 we added all variables that are provided in the data set from the challenge to either path state information h_p or link state information h_l. When referring to such variables, we will provide the names of the variables as named in the data sets in parenthesis to make cross-referencing the source code easier. As the dimension is still greater than the number of variables, all unused components of h_p and h_l are again initialized with 0. To be precise, we added link capacity (`bandwidth`), the scheduling policy (`schedulingPolicy`) and weights for scheduling as link information. As there are three different ToS,

there are also three weights for the policies WFQ and DRR. For the policy SP, we artificially set these three weights to 1.

For the scheduling policy, we used dummy variables, since there are three different policies. Let $\mathbf{e}_i = (e_{i1}, e_{i2}, e_{i3})' \in \mathbb{R}^3$ be the ith canonical vector with $e_{ij} = 1$ if $i = j$ and $e_{ij} = 0$ otherwise. Note that e_i' denotes the transpose of e_i. For simplicity, the scheduling policy s is identified as integers 0, 1 or 2. Then the dummy variable for the scheduling policy can be written as e_{s+1} (sometimes known as "one hot" encoding).

For this particular data set there is exactly one flow for each path as already mentioned in Section 3. Hence, we can identify paths with flows and can therefore assign each path a ToS. That is why we can use ToS as path information. Other variables for the path information are the average data rate on that path (`AvgBw`), the generated packets (`PKtsGen`), average bit rate per time unit (`EqLambda`), average number of packets of average size generated per time unit (`AvgPktsLambda`), information about packet sizes (`AvgPktSize`, `PktSize1`, `PktSize2`) and a variable describing the upper limit for the inter-packet arrival times used in the OMNeT++ simulation (`ExpMaxFactor`). All these variables were as well shifted into $[0, 1]$ to improve the stability of the model.

We decided to split the average desired data rate on a path (`AvgBw`) into different variables for each ToS, respectively. For example, if the ToS is 1, then the first of these three variables contains the average data rate, while the other two are set to 0. For illustration, let $d \in \mathbb{R}_{\geq 0}$, $t \in \{0, 1, 2\}$ be the data rate and ToS, where the ToS is identified by integers. Then this data rate dummy variable can be written as $d \cdot \mathbf{e}_{t+1}$. We also used ToS additionally for the initial path state information. It should be noted that many of those variables listed above are highly correlated. However, we did not encounter any problems and decided to keep these variables without any further modification. By adding these additional variables, we now take into account the scheduling and therefore the prediction of average delays improved significantly.

For illustration, the state information are given by

$$h_l = [x, w_1, w_2, w_3, e'_{s+1}, 0, \dots, 0]' \in \mathbb{R}^{32} \text{ and}$$
$$h_p = [z \cdot e'_{t+1}, \dots, 0]' \in \mathbb{R}^{32},$$

where x denotes the link capacity, w_i ($i = 1, 2, 3$) for the weights, s for the scheduling policy, t for the ToS and z for the average path data rate.

Note that some variables are node properties in the data set, for example the queue scheduling policy that is used. However, flows have a direction. Let us consider a flow on the link from node A to node B. Then we assign this link the scheduling policy from the source node A. Conversely, if we have a flow in the opposite direction on the link from node B to node A, then we assign the scheduling policy from node B. Although both links connect the same nodes, they are treated as different links.

Adding these variables improves the model as scheduling information is now used as input. The average error is about 4.46% (95% CI [3.97%, 4.94%]) as can be seen under Step 3 in Table 1.

4.4 Residual connection

For the readout neural network, we used a similar idea already used in the original RouteNet model [3]. They used a residual connection for the path information to the last hidden layer of the readout neural network. This can be seen as some kind of residual neural network [21]. However, this idea is not present in the RouteNet code provided for the challenge. The readout neural network consists of two hidden layers. The output of this neural network together with the final path state information is used as input in a second neural network with one hidden layer and without any activation function (which is equivalent to a linear activation function) as the path state information can be important for estimating the average delays. The number of neurons for this layer is chosen to be equal to the dimension of the input.

The results are similar to the earlier results. The average error for Step 4 is about 4.55% (95% CI [4.38%, 4.71%]). However, the standard deviation is reduced by a factor of about $3 = 0.39/0.13$, which means the results are stabler, which can be explained by this residual neural network. There are hypotheses that such neural networks smooth the loss function and the algorithm does get stuck less often in non-optimal local minima [21][22].

To illustrate this modification, we refer to the pseudo code 2. In contrast to the unmodified code 1, the readout neural network is separated into two feed forward neural networks. The output of the first neural network with two hidden layers and "relu" activation functions is used as input for the second neural network. Note that the path state information is used in both neural networks as input.

Data: path state h_p and link state vector h_l
Result: predicted per-path delay $\hat{y}_p$
for *t = 0 to T* **do**
  $H_p^{t+1} = G_p(h_p^t, h_l^t)$
  $h_l^{t+1} = G_l(f(H_p^{t+1}), h_l^t)$
  $h_p^{t+1} = g(H_p^{t+1})$
end
$r = R_1(h_p^T)$
$\hat{y}_p = R_2(r, h_p^T)$

Algorithm 2: RouteNet architecture with modified readout neural network

4.5 Stacked gated recurrent networks

The idea of the RouteNet architecture is that for each path/flow we have information about all links of which the path consists. And this link information is used as input in a gated recurrent neural network. The initial infor-

mation is the current path state. For the first cell of the gated recurrent unit (GRU), the information of the first link of a path is being used as the input. Then, the result of this first calculation, and the information of the second link is used in the next step. This is done until all link information of a path has been used. This works well as long as no heterogeneous queuing is in the data. However, to tackle the additional complexity of queuing, we decided to use two gated recurrent networks stacked together. The idea of using stacked gated recurrent networks has already been used in a slightly different context [23], where neural networks predicted traffic volume in road networks to relieve traffic congestion. Furthermore, these gated recurrent networks seemingly allow for more flexibility as more parameters can be trained. The average error for this Step 5 is about 3.18% (95% CI [2.94%, 3.41%]); see Table 1.

Data: path state $h_{p,1}$, $h_{p,2}$ and link state vector h_l
Result: predicted per-path delay $\hat{y}_p$
for *t = 0 to T* **do**
  $H_p^{t+1} = G_p(h_{p,1}^t, h_{p,2}^t, h_l^t)$
  $h_l^{t+1} = G_l(f(H_p^{t+1}), h_l^t)$
  $h_{p,1}^t = g_1(H_p^{t+1})$
  $h_{p,2}^t = g_1(H_p^{t+1})$
end
$r = R_1(h_{p,1}^T, h_{p,2}^T)$
$\hat{y}_p = R_2(r, h_{p,1}^T, h_{p,2}^T)$

Algorithm 3: RouteNet architecture with stacked gated recurrent networks. Each GRN has its own hidden states denotes by $h_{p,i}$, $i = 1, 2$. The functions g_1 and g_2 return the final hidden state for each gated recurrent network.

4.6 Dimension path and link information

As the problem of prediction average delays in networks with scheduling is more complex than without scheduling, it may be necessary to have a higher dimension of path and link state information. The RouteNet code initially used a dimension of 32 for both. We tried increasing the dimension to 64, 128, and 256. For a dimension of 64, we observe a significant increase of the overall error over just using a dimension of 32. Doubling the dimension to 128 again reduces the prediction error significantly. However, using dimension 256 seems to increase the error, which may be a result of overfitting. For this setting, we did not try to add more regularization to avoid a possible overfit. But rather, we decided to set the dimension to 128 in the following. Another reason is computational complexity as we want to train the model in a reasonable amount of time. The results are given again in Table 1 where Step 6A represents dimension 64 with an average error of 2.02%, 6B with a dimension of 128 and an average error of 1.6% and 6C with a dimension of 256 and an error of 2.86%.

4.7 Neurons

The final path state information is obtained through the message passing [2] loop. This final information is mainly used for predicting the average delays in two steps, one neural network with two hidden layers each with 8 neurons. The other neural network does not contain an activation function and is described in Step 4.4. As we have increased the dimension of this path state information in the previous step from 32 to 128, it may be useful to increase the number of neurons in the first neural network responsible for prediction. The baseline number of neurons is 8. We increased this number to 128 and 256 and observed that there is no significant difference between 8, 128 or 256 neurons. The results for 128 and 256 neurons are given in Table 1 as Step 7A and Step 7B with an average error of 1.61% and 1.67%, respectively. As already mentioned, there is no difference to 8 neurons under Step 6B with an error of 1.6%. As the standard deviation of the results for 128 neurons (0.047) seems to be smaller than for just 8 neurons (0.061), we decided to include this change in our final solution. But as this decision is based on only 5 observations, it is not conclusive.

4.8 Decay rate

For the two final steps, we want to optimize this algorithm with respect to learning parameters. We tried two different approaches. First, we usually trained the models for 600 000 steps. For each 60 000 steps, the learning rate is decreased exponentially with a decay rate of 0.6. That is, the current learning rate r_n after n training steps is given by $r_n = 0.001 \cdot 0.6^{\lfloor n/60\,000 \rfloor}$ where $\lfloor . \rfloor$ denotes the floor function. This means that after 600 000 steps the learning rate is almost zero and no changes are observed anymore to the parameters. That is why we decided to increase the learning rate again after 600 000 steps artificially by changing the decay rate to 0.85. Then, the adjusted learning rate is given by $r_n = 0.001 \cdot 0.85^{\lfloor n/60\,000 \rfloor}$ for $n \geq 600\,000$. We refer to this approach as Step 8A and it is related to the concept of cyclical learning rates [24] where the learning rate is increasing and decreasing in a cyclical way.

We compared this approach where we change the decay rate in the beginning of the training to 0.85. We call this approach Step 8B. The former method returns an average error of about 1.47%, the latter an average error of 1.36% as can be seen in Table 1. A graphical representation of the loss functions up to 1.2 million training steps is given in Figure 3 and Figure 4. In Figure 3, there is an increase in the loss function after 600 000 steps as the learning rate was modified at that point. Note that for both loss functions the mean absolute percentage errors are shown shown on a log scale.

As no overfitting was observed we did not change the regularization and decided to keep the standard parameters from RouteNet. In our tests, method B where we changed the decay rate in the beginning performed slightly better.

Table 1 – Mean absolute percentage error (MAPE) for each modification step for five runs: Last two columns denote the average and standard deviation of each row.

	Step	Run 1	Run 2	Run 3	Run 4	Run 5	$\hat{\mu}$	$\hat{\sigma}$
0	MSE	337	120	335	102	185	216	114
1*	MAPE	26.4	64.1	43.7	36.9	60.9	46.4	16.0
2*	Normalization	23.7	23.7	23.7	23.7	23.7	23.7	0.01
3*	Variables	4.55	4.85	4.58	3.80	4.51	4.47	0.39
4*	Residual connection	4.45	4.75	4.53	4.41	4.59	4.55	0.13
5*	Stacked GRN	3.05	3.32	3.40	3.17	2.94	3.18	0.19
6A	Dimension path and link state (64)	2.03	1.94	1.97	1.86	2.28	2.02	0.16
6B*	Dimension path and link state (128)	1.68	1.63	1.52	1.58	1.57	1.60	0.06
6C	Dimension path and link state (256)	2.65	2.99	3.00	3.19	2.48	2.86	0.29
7A*	Neurons (128)	1.60	1.69	1.59	1.57	1.59	1.61	0.05
7B	Neurons (256)	1.59	1.80	1.71	1.60	1.73	1.67	0.09
8A	Decay rate (0.6/0.85)	1.42	1.61	1.37	1.42	1.52	1.47	0.10
8B*	Decay rate (0.85)	1.35	1.34	1.32	1.49	1.30	1.36	0.08

* Variant selected for final solution

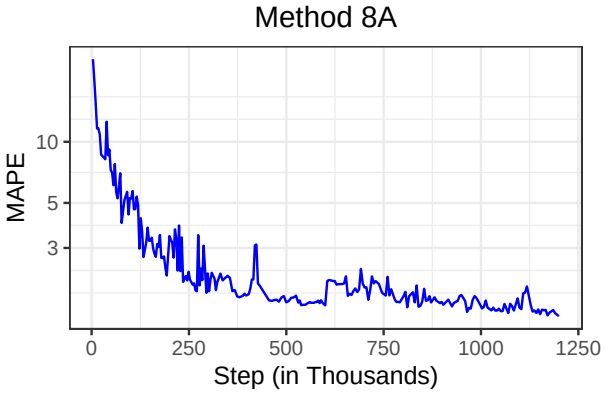


Fig. 3 – Loss function for training during Step 8A

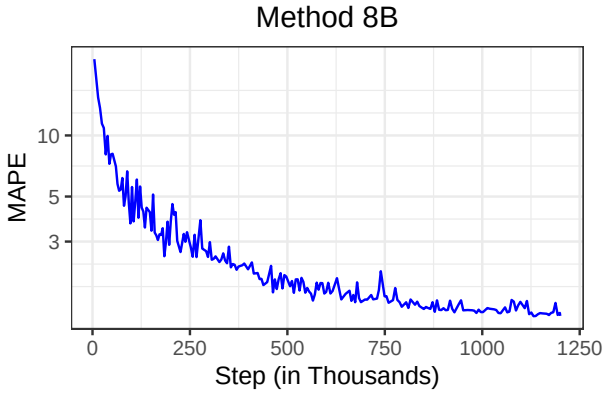


Fig. 4 – Loss function for training during Step 8B

4.9 Final results

Overall, if we evaluate the best trained model, that is run 5 from method 8B as previously described, we get a mean absolute percentage error of about 0.897% with the final data set from the GNN challenge. For comparison, the best result in the challenge by the winning team was an error of 1.53%. Our originally submitted solution achieved an error of about 1.9%, thus we could improve our submitted model further. The training was done on a single Geforce RTX 2080 Ti in under 48 hours for 1.2 million training steps. The code was written in Python 3.7.7 with tensorflow 2.1.0 based on Keras and is available as open source.[1]

5. CONCLUSION

In this paper we have described the problem of estimating delays in communication networks using deep neural networks and proposed a solution based on the RouteNet model [3]. We decomposed our solution into several steps to demonstrate the improvement of each step and compared different variants of the steps to find good hyperparameters. Such a step by step analysis of changes can be helpful in constructing, improving and understanding a model. Using this approach we were able to obtain an error of about 0.897% for predicting average per-path delays based on graph neural networks.

ACKNOWLEDGMENTS

The research was supported by the WISS 2025 (Science and Innovation Strategy Salzburg 2025) project "IDALab Salzburg" (20204-WISS/225/197-2019 and 20102-F1901166-KZP) and the 5G-AI-MLab by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) and the Austrian state Salzburg.

[1] `https://github.com/ITU-AI-ML-in-5G-Challenge/GNN_Challenge_SalzburgResearch_Follow_Up_Paper`

REFERENCES

[1] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.

[2] Krzysztof Rusek and Piotr Chołda. "Message-passing Neural Networks Learn Little's Law". In: *IEEE Communications Letters* 23.2 (2018), pp. 274–277.

[3] Krzysztof Rusek, José Suárez-Varela, Albert Mestres, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "Unveiling the Potential of Graph Neural Networks for Network Modeling and Optimization in SDN". In: *Proceedings of the ACM Symposium on SDN Research*. 2019, pp. 140–151.

[4] Krzysztof Rusek, José Suárez-Varela, Paul Almasan, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN". In: *IEEE Journal on Selected Areas in Communications* 38.10 (2020), pp. 2260–2270.

[5] András Varga and Rudolf Hornig. "An Overview of the OMNeT++ Simulation Environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. 2008, p. 60.

[6] Robert B Cooper. "Queueing Theory". In: *Proceedings of the ACM'81 Conference*. 1981, pp. 119–122.

[7] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Vol. 2050. Springer Science & Business Media, 2001.

[8] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities". In: *Journal of Internet Services and Applications* 9.1 (2018), pp. 1–99.

[9] Albert Mestres, Eduard Alarcón, Yusheng Ji, and Albert Cabellos-Aparicio. "Understanding the Modeling of Computer Network Delays Using Neural Networks". In: *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. 2018, pp. 46–52.

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural Message Passing for Quantum Chemistry". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1263–1272.

[11] Fabien Geyer and Georg Carle. "Learning and Generating Distributed Routing Protocols using Graph-Based Deep Learning". In: *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. 2018, pp. 40–45.

[12] Miquel Ferriol-Galmés, José Suárez-Varela, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "Applying Graph-Based Deep Learning to Realistic Network Scenarios". In: *arXiv preprint arXiv:2010.06686* (2020).

[13] Barcelona Neural Networking Center. *GNN Challenge Website*. `https://bnn.upc.edu/challenge2020/`. Accessed: 2021-01-22.

[14] Xiaojun Hei, Jun Zhang, Brahim Bensaou, and Chi-Chung Cheung. "Wavelength Converter Placement in Least-Load-Routing-Based Optical Networks Using Genetic Algorithms". In: *Journal of Optical Networking* 3.5 (2004), pp. 363–378.

[15] Fernando Barreto, Emílio CG Wille, and Luiz Nacamura Jr. "Fast Emergency Paths Schema to Overcome Transient Link Failures in OSPF Routing". In: *arXiv preprint arXiv:1204.2465* (2012).

[16] João Pedro, João Santos, and João Pires. "Performance Evaluation of Integrated OTN/DWDM Networks with Single-Stage Multiplexing of Optical Channel Data Units". In: *13th International Conference on Transparent Optical Networks*. IEEE. 2011, pp. 1–4.

[17] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach, 7th Edition*. Pearson Education Limited, 2017. ISBN: 978-0-13-359414-0.

[18] Knowledge-Defined Networking. *RouteNet Challenge Github Repository*. `https://github.com/knowledgedefinednetworking/RouteNet-challenge`. Accessed: 2021-02-03.

[19] Jorge Sola and Joaquin Sevilla. "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems". In: *IEEE Transactions on Nuclear Science* 44.3 (1997), pp. 1464–1468.

[20] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. "Efficient Backprop". In: *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 9–48.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity Mappings in Deep Residual Networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[23] Peng Sun, Azzedine Boukerche, and Yanjie Tao. "SSGRU: A Novel Hybrid Stacked GRU-Based Traffic Volume Prediction Approach in a Road Network". In: *Computer Communications* 160 (2020), pp. 502–511.

[24] Leslie N Smith. "Cyclical Learning Rates for Training Neural Networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

AUTHORS

Martin Happ received his Ph.D. in Mathematics at the University of Salzburg in 2020. His research interests include multivariate statistics, especially repeated measurements, and nonparametric statistics with a focus on rank procedures. Recently, he started working on machine learning with application in communication networks in his current position at Salzburg Research.

Matthias Herlich received a doctorate degree in Computer Science from the Universität Paderborn in Germany. After working at the Palo Alto Research Center and the National Institute of Informatics in Tokyo he went to Salzburg Research. His experience ranges from analytic and simulation-based approaches to evaluation of measurements. Recently, he has begun to apply his knowledge to machine learning for communication networks.

Christian Maier received a BSc in Mathematics from the Technical University of Munich. He is now a researcher at Salzburg Research. His work focuses on reliability and bandwidth measurements in wireless networks, and on the application of machine learning to communication technologies.

Jia Lei Du received his doctorate degree (Dr.-Ing.) in Electrical Engineering from the Heinz Nixdorf Institute, University of Paderborn, Germany for the study of mobile robot cooperation and communication systems, and his diploma degree (Dipl.-Ing.) in Engineering Cybernetics from the University of Stuttgart, Germany. Jia Lei Du also worked as a researcher at Philips Medical Systems, Boeblingen, Germany where he was responsible for a key component for the reliable communication in a new patient monitoring product. He is now a researcher at Salzburg Research and his interests include communication networks, distributed systems, and multi-robot systems.

Peter Dorfinger received his diploma degree (DI(FH)) from the Department of Telecommunications Engineering at the Salzburg University of Applied Sciences in 2002. In 2010 he received his master's degree (DI) from the Department of Information Technologies and Systems Management at the Salzburg University of Applied Sciences. He is head of group, senior researcher and project manager within Salzburg Research. He has published more than 40 research papers. His research focus is on wired and wireless critical infrastructure networks with a specific focus on measurement and monitoring. Since 2007, he lectures on network protocols and services at the Salzburg University of Applied Sciences.

SITE-SPECIFIC MILLIMETER-WAVE COMPRESSIVE CHANNEL ESTIMATION ALGORITHMS WITH HYBRID MIMO ARCHITECTURES

Sai Subramanyam Thoota[1], Dolores Garcia Marti[2,3], Özlem Tuğfe Demir[4,5], Rakesh Mundlamuri[6], Joan Palacios[2,3], Cenk M. Yetis[7], Christo Kurisummoottil Thomas[6], Sameera H. Bharadwaja[1], Emil Björnson[4,5], Pontus Giselsson[7], Marios Kountouris[6], Chandra R. Murthy[1], Nuria González-Prelcic[8], Joerg Widmer[2]
[1]Indian Institute of Science, Bangalore, India, [2]IMDEA Networks, Madrid, Spain, [3]Universidad Carlos III, Madrid, Spain, [4]KTH Royal Institute of Technology, Stockholm, Sweden, [5]Linköping University, Linköping, Sweden, [6]EURECOM, Sophia-Antipolis, France, [7]Lund University, Lund, Sweden, [8]North Carolina State University, Raleigh, USA

NOTE: Corresponding author: Sai Subramanyam Thoota, thoota@iisc.ac.in

Abstract – *In this paper, we present and compare three novel model-cum-data-driven channel estimation procedures in a millimeter-wave Multi-Input Multi-Output (MIMO) Orthogonal Frequency Division Multiplexing (OFDM) wireless communication system. The transceivers employ a hybrid analog-digital architecture. We adapt techniques from a wide range of signal processing methods, such as detection and estimation theories, compressed sensing, and Bayesian inference, to learn the unknown virtual beamspace domain dictionary, as well as the delay-and-beamspace sparse channel. We train the model-based algorithms with a site-specific training dataset generated using a realistic ray tracing-based wireless channel simulation tool. We assess the performance of the proposed channel estimation algorithms with the same site's test data. We benchmark the performance of our novel procedures in terms of normalized mean squared error against an existing fast greedy method and empirically show that model-based approaches combined with data-driven customization unanimously outperform the state-of-the-art techniques by a large margin. The proposed algorithms were selected as the top three solutions in the "ML5G-PHY Channel Estimation Global Challenge 2020" organized by the International Telecommunication Union.*

Keywords – Bayesian inference, channel estimation, compressed sensing, data-driven, hybrid MIMO, mmWave

1. INTRODUCTION

Millimeter-Wave (mmWave) wireless communication is one of the potential technologies proposed for the next generation communication systems (5G and beyond) to meet the ever-increasing demand for high data rates. The mmWave frequency spectrum, ranging from 30 GHz to 300 GHz, is attractive because it offers large bandwidths ($\sim$ 2GHz), resulting in very high data rates and low latency. These advantages come at a cost of higher path loss due to several factors, such as blockages and oxygen absorption at mmWave frequencies, which in turn bring several engineering challenges in adopting this technology in commercial wireless communication systems.

A potential solution to overcome this problem is beamforming, which leverages the availability of multiple antennas at the transmitter and receiver. In particular, millimeter wavelengths enable one to accommodate a larger number of antennas into the same physical space, and thereby attain high beamforming gains. However, a fully digital architecture in a Multi-Input Multi-Output (MIMO) system, i.e., one Radio Frequency (RF) chain per antenna, and one complex-valued Analog-to-Digital Converter (ADC) per RF chain is less appealing both from commercial and engineering perspectives due to its high cost and energy requirements. Therefore, a hybrid MIMO architecture is proposed in the literature as a potential solution to solve this problem [1].

In a hybrid MIMO system, multiple antennas are connected to an RF chain using a phase shifter network (RF precoder/combiner), and a digital precoder/combiner is employed in the complex baseband side of the transceiver. The RF and digital precoders/combiners are configured by optimizing a system performance metric such as the sum rate or signal to interference noise ratio. Unlike a fully analog architecture, a hybrid architecture allows one to reduce the number of RF chains, while supporting multi-stream and multi-user transmissions. The major challenges then are in estimating the mmWave wireless channel and configuring the RF and digital precoders/combiners based on the channel estimate. The problem is exacerbated by the fact that only the low dimensional RF combined signals at the baseband are available for estimating the channel. Since the system does not have any knowledge of the channel state during the channel estimation phase, the baseband precoders/combiners are set to the identity matrix and random phase shifts are chosen for the RF precoders/combiners.

MmWave channel estimation in a hybrid MIMO architecture is a well studied problem, and we provide a brief overview of some of the key existing literature here. The simplest channel estimation method in hybrid MIMO systems is the Least Squares (LS)-based approach [2], which is inherited from conventional MIMO [3]. A more refined solution to channel estimation is to exploit both the delay

and angular domain sparsity that mmWave channels exhibit. In this approach, the channel estimation problem is formulated as a sparse recovery problem [4]. Such compressive sensing based estimation techniques were first developed for frequency-flat hybrid mmWave MIMO systems [5, 6]. Recently, frequency-selective channels with OFDM-based communications leading to a more complex estimation problem have also been considered, with different approaches to exploit the sparse channel characteristics [4, 7, 8]. Several model-based signal processing techniques for mmWave channel estimation under various system settings can be found in [9–23].

Machine Learning and Artificial Intelligence (ML/AI) have been shown to be powerful tools in diverse areas such as natural language processing, speech processing, and image recognition, where it is challenging to design specific model-based algorithms. However, the impact of ML/AI on the design and optimization of communication systems is yet to be extensively studied, especially under realistic and practically meaningful settings. We aim to address some of the aspects of ML/AI in wireless communications here.

In this paper, we study the potential advantage of using data-driven approaches for channel estimation in hybrid MIMO systems. The model-cum-data driven algorithms we develop in this paper were selected as the **top three** solutions in the "**ML5G-PHY Channel Estimation Global Challenge 2020**" organized by the International Telecommunication Union (ITU)[1]. Our main goal in this paper is to present and contrast these three algorithms for estimating an mmWave channel in a hybrid MIMO system. We compare the Normalized Mean Squared Error (NMSE) performance of these approaches and discuss the machine learning techniques relevant for the challenge at hand. These approaches utilize the channel training datasets generated using the *Raymobtime* tool to customize the algorithms so that they perform well for a test dataset generated in a similar environment [24].

We provide a brief overview of the three solutions below:

1. We integrate a fast greedy search with a high-performing Bayesian inference method in the first approach.[2] We use a Multi-Level Greedy Search (MLGS) to learn the sparsifying virtual beamspace dictionary that reduces the dimensionality of the problem and use the learned dictionary to estimate the channel using a Sparse Bayesian Learning (SBL) method. We finally exploit the delay-domain sparsity to de-noise the estimated channels. We name the algorithm as MLGS-SBL.

2. As a second approach, we propose another SBL-based algorithm to exploit the sparsity of the channel. We utilize the pattern-coupling concept to model possible block sparsity patterns among the consecutive Angle Of Arrivals (AoAs) and Angle Of Departures (AoDs). As a first step, we obtain the time-domain channels from the provided training dataset via the inverse Discrete Fourier Transform (DFT) and remove the channel taps with small magnitude. Then, we apply the algorithm to the ground truth time-domain channels to obtain the sparse representations. Using joint angular distribution learned from training data, we refine the grids and pattern-coupling relations in the testing stage to improve the channel estimation quality. This approach is called "Pattern-Coupled Sparse Bayesian Learning for Channel Estimation with Dominating Delay Taps (PCSBL-DDT)" in the paper.

3. The third approach, Projection Cuts Orthogonal Matching Pursuit (PC-OMP), is based on the Orthogonal Matching Pursuit (OMP) algorithm. This method makes use of the sparsity of the mm-wave channel to extract channel components. At each iteration of the OMP algorithm, a coarse estimate of the strongest path parameters (AoA, AoD, and delay) is obtained by a low resolution grid search. Then, each of the three parameters is refined alternately, assuming the other two to be known. In this way, we keep the algorithm's complexity low without compromising on its accuracy. At the end of each iteration, a path detection hypothesis is tested, and, if successful, the path is subtracted from the channel. This process is repeated until no additional path is detected.

1.1 Notation

The operator $(\cdot)^*$ represents the conjugate transpose or conjugate for a matrix or a scalar, respectively. $\bar{\mathbf{A}}$, $\mathbf{A}^T$, and $\mathbf{A}^\dagger$ denote the conjugate, transpose, and Moore-Penrose pseudoinverse of a matrix $\mathbf{A}$, respectively. The multivariate complex Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$ is denoted by $\mathcal{CN}(\boldsymbol{\mu}, \mathbf{C})$ and its probability density function (pdf) of a random vector $\mathbf{x}$ is denoted by $\mathcal{CN}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C})$. blkdiag$(\cdot)$ represents the block-diagonal part of a matrix. diag$(\mathbf{X})$ or diag$(\mathbf{x})$ represents a vector obtained by the diagonal elements of the matrix $\mathbf{X}$ or the diagonal matrix obtained with the elements of $\mathbf{x}$ in the diagonal, respectively. $\mathbf{A} \otimes \mathbf{B}$ denotes the Kronecker product of the matrices $\mathbf{A}$ and $\mathbf{B}$. $||\mathbf{A}||_F$ denotes the Frobenius norm of a matrix $\mathbf{A}$. $\langle \mathbf{a}, \mathbf{b} \rangle$ is the inner product of the two vectors $\mathbf{a}$ and $\mathbf{b}$. The trace of a matrix $\mathbf{A}$ is denoted by tr$(\mathbf{A})$. Tx and Rx denote the transmitter and receiver, respectively. We use $vec(\mathbf{A})$ to vectorize the matrix $\mathbf{A}$ column-wise. $\mathbb{E}[\cdot]$ denotes the expectation.

2. SYSTEM MODEL

We consider a single cell mmWave hybrid MIMO-OFDM system with N_t antennas at the transmitter (Tx) and N_r antennas at the receiver (Rx), as shown in Fig. 1.

[1] https://www.itu.int/en/ITU-T/AI/challenge/2020/Pages/default.aspx

[2] The order in which the algorithms are presented is unrelated to their ranking in the ITU ML5G-PHY channel estimation challenge. The ordering is based on ease of presentation and readability of the paper.

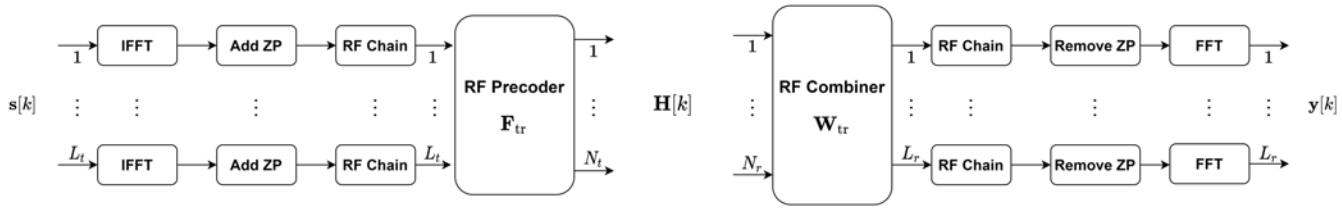


Fig. 1 – mmWave MIMO system based on a hybrid analog-digital architecture.

The Tx and Rx are equipped with L_t and L_r RF chains, respectively. The training input signal $\mathbf{s}[k] \in \mathbb{C}^{L_t \times 1}$ on the k^{th} subcarrier is OFDM modulated, up-converted to RF, and analog precoded using $\mathbf{F}_{\text{tr}} \in \mathbb{C}^{N_t \times L_t}$, and transmitted over the air to the Rx via an mmWave channel denoted by $\mathbf{H}[k]$ on the k^{th} subcarrier. The received signal is filtered using an RF combining matrix $\mathbf{W}_{\text{tr}} \in \mathbb{C}^{N_r \times L_r}$, down-converted to baseband, OFDM demodulated to obtain the k^{th} subcarrier's complex baseband signal $\mathbf{y}[k] \in \mathbb{C}^{L_r \times 1}$. We denote the total number of subcarriers by K.

In the initial access phase, the system has no prior knowledge of the channel, and therefore the precoder and combiner matrices cannot be designed to optimize any chosen performance metric. Hence, we choose random analog precoding and combining matrices (with unit modulus entries). In our system model, we adopt a fully connected phase shifter network for analog precoding/combining. The analog precoders and combiners are frequency-flat, and thus are the same for each subcarrier $k = 1, \dots, K$. The system operates with Uniform Linear Arrays (ULAs) at both the Tx and Rx with half wavelength spacing be-tween consecutive antennas. The total number of training frames is denoted by M.

After RF combining, down-conversion, zero prefix removal and DFT, the complex baseband signal received during the m^{th} training frame for the k^{th} subcarrier, denoted by $\mathbf{y}^{(m)}[k] \in \mathbb{C}^{L_r \times 1}$ is given by

$$\mathbf{y}^{(m)}[k] = \mathbf{W}_{\text{tr}}^{(m)^*}(\mathbf{H}[k]\mathbf{F}_{\text{tr}}^{(m)}\mathbf{q}^{(m)}t^{(m)}[k] + \mathbf{n}^{(m)}[k]), \quad (1)$$

for $m = 1, \dots, M$ where $\mathbf{H}[k] \in \mathbb{C}^{N_r \times N_t}$ represents the frequency domain MIMO channel matrix for the k^{th} subcarrier. We choose the m^{th} training signal as $\mathbf{s}^{(m)}[k] = \mathbf{q}^{(m)}t^{(m)}[k]$, where $t^{(m)}[k] \in \mathbb{C}$ is a subcarrier-dependent pilot symbol, and $\mathbf{q}^{(m)} \in \mathbb{C}^{L_t \times 1}$ is a frequency-flat vector whose entries are chosen as $\frac{1}{\sqrt{2L_t}}(a + jb)$, where $a, b \in \{-1, 1\}$ and are uniformly distributed. The noise vector $\mathbf{n}^{(m)}[k]$ is independently and identically distributed across K subcarriers as $\mathcal{CN}(\mathbf{0}, \sigma^2\mathbf{I}_{N_r})$. We define the transmit Signal-to-Noise Ratio (SNR) as $\rho = \frac{1}{\sigma_n^2}$. After compensating for $t^{(m)}[k]$, and vectorizing *(1)*, we use the result $vec(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})vec(\mathbf{X})$ to obtain

$$vec(\mathbf{y}^{(m)}[k]) = \underbrace{\left(\mathbf{q}^{(m)\,T}\mathbf{F}_{\text{tr}}^{(m)\,T} \otimes \mathbf{W}_{\text{tr}}^{(m)^*}\right)}_{\mathbf{\Phi}^{(m)}} vec(\mathbf{H}[k]) + \mathbf{W}_{\text{tr}}^{(m)^*}\mathbf{n}^{(m)}[k]. \quad (2)$$

Next, we describe the mmWave channel model.

2.1 Channel model

We consider a frequency-selective geometric channel model that is constant across M training frames, and has N_c delay taps [4, 25]. The d^{th} delay tap is modeled as a clustered channel with L paths as

$$\mathbf{H}_d = \sqrt{\frac{N_t N_r}{L\rho_L}} \sum_{\ell=1}^{L} \alpha_\ell p(dT_s - \tau_\ell)\mathbf{a}_{\text{R}}(\phi_\ell)\mathbf{a}_{\text{T}}^*(\theta_\ell), \quad (3)$$

where ρ_L is the path loss between Tx and Rx, α_ℓ represents the complex path gain, ϕ_ℓ is the AoA, θ_ℓ is the AoD, τ_ℓ denotes the delay of the ℓ^{th} path. The corresponding Rx and Tx array steering vectors are denoted by $\mathbf{a}_{\text{R}}(\phi_\ell) \in \mathbb{C}^{N_r \times 1}$ and $\mathbf{a}_{\text{T}}(\theta_\ell) \in \mathbb{C}^{N_t \times 1}$, respectively. The pulse shaping and other low pass filtering evaluated at τ is represented by $p(\tau)$, and T_s is the sampling interval. We represent the MIMO channel $\mathbf{H}_d$ in a matrix form as

$$\mathbf{H}_d = \mathbf{A}_{\text{R}}\mathbf{\Delta}_d\mathbf{A}_{\text{T}}^*, \quad (4)$$

where $\mathbf{A}_{\text{R}} \in \mathbb{C}^{N_r \times L}$ and $\mathbf{A}_{\text{T}} \in \mathbb{C}^{N_t \times L}$ contain the Rx and Tx array steering vectors $\mathbf{a}_{\text{R}}(\phi_\ell)$ and $\mathbf{a}_{\text{T}}(\theta_\ell)$ as their columns for $\ell = 1, \dots, L$, respectively. $\mathbf{\Delta}_d \in \mathbb{C}^{L \times L}$ is a diagonal matrix containing the complex channel gains. We take a K-point DFT of the delay-domain channel to get the frequency domain representation as

$$\mathbf{H}[k] = \sum_{d=0}^{N_c-1} \mathbf{H}_d \exp\left(-\frac{j2\pi kd}{K}\right) = \mathbf{A}_{\text{R}}\mathbf{\Delta}[k]\mathbf{A}_{\text{T}}^*, \quad (5)$$

for $k = 0, \dots, K-1$, and

$$\mathbf{\Delta}[k] = \sum_{d=0}^{N_c-1} \mathbf{\Delta}_d \exp\left(-\frac{j2\pi kd}{K}\right). \quad (6)$$

We adopt the extended virtual channel model in [25] to represent $\mathbf{H}_d$ as

$$\mathbf{H}_d \approx \tilde{\mathbf{A}}_{\text{R}}\mathbf{\Delta}_d^v\tilde{\mathbf{A}}_{\text{T}}^*, \quad (7)$$

where the dictionary matrices $\tilde{\mathbf{A}}_{\mathrm{R}} \in \mathbb{C}^{N_r \times G_r}$ and $\tilde{\mathbf{A}}_{\mathrm{T}} \in \mathbb{C}^{N_t \times G_t}$ contain the Rx and Tx array steering vectors evaluated on a grid of size G_r for the AoA and a grid of size G_t for the AoD, respectively. When G_r and G_t are chosen properly, i.e., much greater than L, $\boldsymbol{\Delta}_d^v \in \mathbb{C}^{G_r \times G_t}$ becomes a sparse matrix containing the channel path gains on the locations that match with the actual AoDs and AoAs. We represent *(7)* in the frequency domain as

$$\mathbf{H}[k] \approx \tilde{\mathbf{A}}_{\mathrm{R}} \boldsymbol{\Delta}^v[k] \tilde{\mathbf{A}}_{\mathrm{T}}^*, \tag{8}$$

for $k = 0, \dots, K-1$, and

$$\boldsymbol{\Delta}^v[k] = \sum_{d=0}^{N_c-1} \boldsymbol{\Delta}_d^v \exp\left(-\frac{j2\pi kd}{K}\right). \tag{9}$$

Note that the dictionary matrices $\tilde{\mathbf{A}}_{\mathrm{R}}$ and $\tilde{\mathbf{A}}_{\mathrm{T}}$ are common to all the subcarriers due to the frequency-flat array response vectors. Hence, the sparse matrices $\boldsymbol{\Delta}^v[k]$ for $k = 1, \dots, K$ have the non-zero elements at the same indices. This means that they share a common sparsity pattern [4].

Now, we vectorize *(8)* to get

$$vec(\mathbf{H}[k]) = \left(\bar{\tilde{\mathbf{A}}}_{\mathrm{T}} \otimes \tilde{\mathbf{A}}_{\mathrm{R}}\right) vec(\boldsymbol{\Delta}^{\mathrm{v}}[k]). \tag{10}$$

We define $\boldsymbol{\Psi} = \bar{\tilde{\mathbf{A}}}_{\mathrm{T}} \otimes \tilde{\mathbf{A}}_{\mathrm{R}} \in \mathbb{C}^{N_t N_r \times G_t G_r}$ and $\mathbf{h}^{\mathrm{v}}[k] = vec(\boldsymbol{\Delta}^{\mathrm{v}}[k]) \in \mathbb{C}^{G_t G_r}$, and substitute $vec(\mathbf{H}[k])$ in *(2)* to get

$$vec(\mathbf{y}^{(m)}[k]) = \boldsymbol{\Phi}^{(m)} \boldsymbol{\Psi} \mathbf{h}[k] + \mathbf{n}_c^{(m)}[k], \tag{11}$$

where $\mathbf{n}_c^{(m)}[k] = \mathbf{W}_{\mathrm{tr}}^{(m)^*} \mathbf{n}^{(m)}[k]$. By concatenating the RF combined signals of M training frames, we get

$$\underbrace{\begin{bmatrix} \mathbf{y}^{(1)}[k] \\ \vdots \\ \mathbf{y}^{(M)}[k] \end{bmatrix}}_{\mathbf{y}[k]} = \underbrace{\begin{bmatrix} \boldsymbol{\Phi}^{(1)} \\ \vdots \\ \boldsymbol{\Phi}^{(M)} \end{bmatrix}}_{\boldsymbol{\Phi}} \boldsymbol{\Psi} \mathbf{h}^{\mathrm{v}}[k] + \underbrace{\begin{bmatrix} \mathbf{n}_c^{(1)}[k] \\ \vdots \\ \mathbf{n}_c^{(M)}[k] \end{bmatrix}}_{\mathbf{n}_c[k]}. \tag{12}$$

Now, by stacking the received signals of K subcarriers, we get the final system equation

$$\begin{aligned} \mathbf{Y} &= \begin{bmatrix} \mathbf{y}[1] & \dots & \mathbf{y}[K] \end{bmatrix} \\ &= \boldsymbol{\Phi}\boldsymbol{\Psi} \begin{bmatrix} \mathbf{h}^{\mathrm{v}}[1] & \dots & \mathbf{h}^{\mathrm{v}}[K] \end{bmatrix} + \begin{bmatrix} \mathbf{n}_c[1] & \dots & \mathbf{n}_c[K] \end{bmatrix} \\ &= \boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{H}^{\mathrm{v}} + \mathbf{N}_c. \end{aligned} \tag{13}$$

Our goal is to estimate $\mathbf{H}[k]$, for $k = 0, \dots, K-1$, given $\mathbf{Y}$ and $\boldsymbol{\Phi}$. As the AoDs and AoAs are the same for all the subcarriers, $\mathbf{H}^{\mathrm{v}} \in \mathbb{C}^{G_t G_r \times K}$ has a joint row sparse structure, i.e., the support set of each column of $\mathbf{H}^v$ are the same. Also, we do not have the knowledge of the sparsifying dictionary $\boldsymbol{\Psi}$ and the noise variance, which makes the channel estimation problem more challenging. In the following sections, we present three different solutions to this channel estimation problem.

3. MLGS-SBL

In this section, we propose a model-based approach using the framework of Compressed Sensing (CS), to estimate the mmWave channel given the received pilot measurements and the frequency-flat transmit vector. We integrate a fast greedy search procedure and a high performing statistical inference method to estimate the channel. The algorithm consists of the following steps:

1. Preconditioning
2. Multi-level greedy search for dictionary learning
3. Noise variance estimation
4. Sparse Bayesian learning for channel estimation
5. Channel de-noising

We provide a detailed description of each step below.

3.1 Preconditioning

Sparse signal recovery using greedy algorithms, such as OMP, are likely to choose the correct support set when the noise covariance matrix is diagonal. In our mmWave channel estimation problem, RF combining by $\mathbf{W}_{\mathrm{tr}}$ at the front end of the receiver results in correlated noise, which needs to be whitened using a preconditioning filter [4].

The scaled noise covariance matrix before whitening is

$$\begin{aligned} \mathbf{C}_{\mathrm{w}} =& \frac{\mathbb{E}\left[\mathbf{n}_c[k]\mathbf{n}_c^*[k]\right]}{\sigma^2} \\ =& \mathrm{blkdiag}\{\mathbf{W}_{\mathrm{tr}}^{(1)^*}\mathbf{W}_{\mathrm{tr}}^{(1)}, \dots, \mathbf{W}_{\mathrm{tr}}^{(M)^*}\mathbf{W}_{\mathrm{tr}}^{(M)}\}. \end{aligned} \tag{14}$$

We get the above by noting that

$$\mathbb{E}\left[\mathbf{n}_c^{(i)}[k]\mathbf{n}_c^{(j)^*}[k]\right] = \sigma^2 \mathbf{W}_{\mathrm{tr}}^{(i)^*}\mathbf{W}_{\mathrm{tr}}^{(j)}\delta[i-j]. \tag{15}$$

We perform a Cholesky decomposition of $\mathbf{C}_{\mathrm{w}}$ to obtain $\mathbf{C}_{\mathrm{w}} = \mathbf{D}_{\mathrm{w}}^*\mathbf{D}_{\mathrm{w}}$, where $\mathbf{D}_{\mathrm{w}} \in \mathbb{C}^{ML_r \times ML_r}$ is upper triangular. Let us define $\mathbf{D}_{\mathrm{w}}^{-*}$ to denote the inverse of $\mathbf{D}_{\mathrm{w}}^*$. Now, we multiply the RF combined received signal *(12)* by $\mathbf{D}_{\mathrm{w}}^{-*}$ to obtain the noise-whitened received signal:

$$\begin{aligned} \mathbf{y}_{\mathrm{w}}[k] = \mathbf{D}_{\mathrm{w}}^{-*}\mathbf{y}[k] &= \mathbf{D}_{\mathrm{w}}^{-*}\boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{h}^{\mathrm{v}}[k] + \mathbf{D}_{\mathrm{w}}^{-*}\mathbf{n}_c[k] \\ &= \boldsymbol{\Upsilon}_{\mathrm{w}}\mathbf{h}^{\mathrm{v}}[k] + \mathbf{D}_{\mathrm{w}}^{-*}\mathbf{n}_c[k], \end{aligned} \tag{16}$$

where $\boldsymbol{\Upsilon}_{\mathrm{w}} = \mathbf{D}_{\mathrm{w}}^{-*}\boldsymbol{\Phi}\boldsymbol{\Psi} \in \mathbb{C}^{ML_r \times G_t G_r}$. Concatenating the noise-whitened received signals of all the K subcarriers, we get

$$\mathbf{Y}_{\mathrm{w}} = \begin{bmatrix} \mathbf{y}_{\mathrm{w}}[1] & \dots & \mathbf{y}_{\mathrm{w}}[K] \end{bmatrix} = \boldsymbol{\Phi}_{\mathrm{w}}\boldsymbol{\Psi}\mathbf{H}^{\mathrm{v}} + \mathbf{N}_{\mathrm{w}}, \tag{17}$$

where $\mathbf{Y}_{\mathrm{w}} \in \mathbb{C}^{ML_r \times K}$, $\boldsymbol{\Phi}_{\mathrm{w}} = \mathbf{D}_{\mathrm{w}}^{-*}\boldsymbol{\Phi} \in \mathbb{C}^{ML_r \times N_t N_r}$, and $\mathbf{N}_{\mathrm{w}} = \mathbf{D}_{\mathrm{w}}^{-*}\begin{bmatrix} \mathbf{n}[1] & \dots & \mathbf{n}[K] \end{bmatrix} \in \mathbb{C}^{ML_r \times K}$. Thus, we need to estimate the row sparse matrix $\mathbf{H}^{\mathrm{v}}$ given $\mathbf{Y}_{\mathrm{w}}$ and $\boldsymbol{\Phi}_{\mathrm{w}}$.

3.2 Multi-level greedy search

We obtain an initial channel estimate using the MLGS procedure with a coarsely quantized beamspace dictionary. We adopt the Simultaneously Weighted Orthogonal Matching Pursuit (SW-OMP) algorithm as our base algorithm to form an initial estimate of the channel [4]. As the sparsifying dictionary $\mathbf{\Psi}$ is unknown a priori, we use row-truncated DFT matrices of size $N_t \times G_t$ and $N_r \times G_r$ as the Tx and Rx array steering matrices, respectively. Let $\widetilde{\mathbf{\Psi}}$ be the initial sparsifying dictionary.

In the first step of MLGS, we select a column from $\widetilde{\mathbf{\Psi}}$ that is maximally correlated with the received signal. Mathematically,

$$\hat{i} = \arg\max_{i} \sum_{k=1}^{K} \left| \left(\mathbf{\Phi}_{\mathrm{w}} \widetilde{\mathbf{\Psi}}[:, i] \right)^{*} \mathbf{y}_{\mathrm{w}}[k] \right|^{2}, \qquad (18)$$

where $|\cdot|$ denotes an element-wise modulus operation, and $\widetilde{\mathbf{\Psi}}[:, i]$ is the i^{th} column of $\widetilde{\mathbf{\Psi}}$. Once we select $\hat{i}$, we extract AoD $\theta_{\hat{i}}$ and AoA $\phi_{\hat{i}}$ using the structure of $\widetilde{\mathbf{\Psi}}$, and form a finely spaced dictionary of range $(\theta_{\hat{i}} - \Delta\theta, \theta_{\hat{i}} + \Delta\theta)$ and $(\phi_{\hat{i}} - \Delta\phi, \phi_{\hat{i}} + \Delta\phi)$, where $\Delta\theta$ and $\Delta\phi$ are appropriately chosen based on the spatial quantization of the previously chosen dictionary. We repeat *(18)* with $\widetilde{\mathbf{\Psi}}$ replaced by the newly formed dictionary, and choose a new {AoD, AoA} pair. We repeat this process N times and select one set of AoD and AoA. Then, we compute

$$\widehat{\mathbf{H}}^{\mathrm{v}} = \left(\mathbf{\Phi}_{\mathrm{w}} \widehat{\mathbf{\Psi}} \right)^{\dagger} \mathbf{Y}_{\mathrm{w}}, \qquad (19)$$

where $\widehat{\mathbf{\Psi}}$ is formed using the currently chosen AoD and AoA. This whole procedure constitutes the first out of S iterations of the MLGS algorithm in which we recover a single tap.

In the s^{th} iteration of MLGS, we recover s channel taps by following the same steps as above, but with the residual $\mathbf{Y}'_{\mathrm{w}} = \mathbf{Y}_{\mathrm{w}} - \mathbf{\Phi}_{\mathrm{w}} \widehat{\mathbf{\Psi}} \widehat{\mathbf{H}}^{\mathrm{v}}$ as observations, where $\widehat{\mathbf{\Psi}}$ comprises the set of {AoD, AoA} pairs chosen in the first $s-1$ iterations. Therefore, after S iterations, we recover S virtual beamspace channel taps. We summarize MLGS as a flow diagram in Fig. 2.

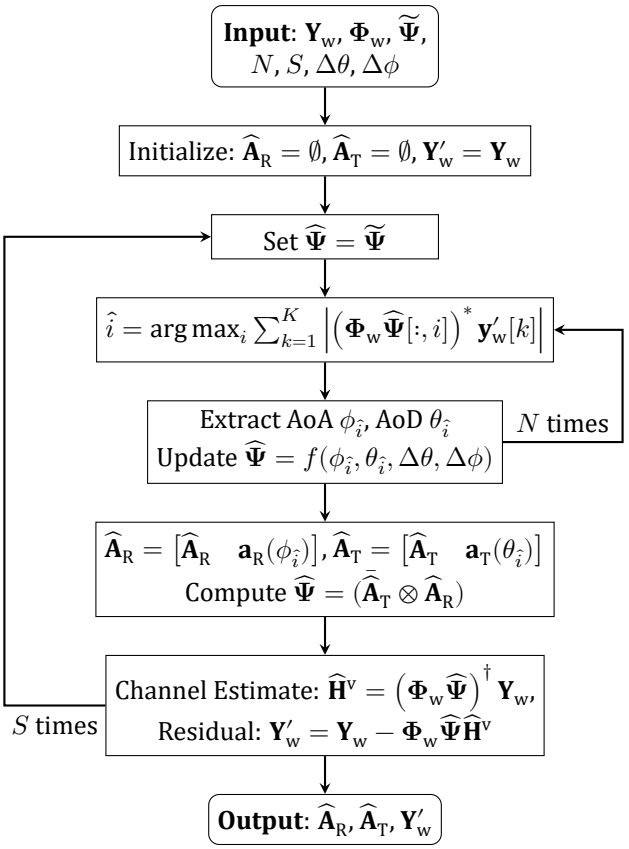


Fig. 2 – Flow diagram of MLGS.

3.3 Noise variance estimation

We estimate the noise variance $\hat{\sigma}_n^2$ using the residual output from MLGS. The noise variance is computed as

$$\hat{\sigma}_n^2 = \frac{1}{MKL_r} ||\mathbf{Y}'_{\mathrm{w}}||_F^2. \qquad (20)$$

3.4 Sparse Bayesian learning

In this step, our goal is to refine the channel estimates output by the MLGS procedure. For convenience, we recall the measurement equation:

$$\mathbf{Y}_{\mathrm{w}} = \mathbf{\Phi}_{\mathrm{w}} \widehat{\mathbf{\Psi}} \mathbf{H}^{\mathrm{v}} + \mathbf{N}_{\mathrm{w}}\,, \qquad (21)$$

where $\widehat{\mathbf{\Psi}} = (\widehat{\mathbf{A}}_{\mathrm{T}} \otimes \widehat{\mathbf{A}}_{\mathrm{R}})$ is the dictionary output by MLGS. We adopt a statistical inference approach to infer the posterior distribution of $\mathbf{H}^{\mathrm{v}}$ given the measurements $\mathbf{Y}_{\mathrm{w}}$, measurement matrix $\mathbf{\Phi}_{\mathrm{w}} \widehat{\mathbf{\Psi}}$, and noise variance $\hat{\sigma}_n^2$.

We use sparse Bayesian learning, a type-II maximum likelihood estimation procedure to obtain the channel estimate [26, 27]. In this method, we consider $\mathbf{H}^{\mathrm{v}}$ as a hidden variable, and obtain its posterior statistics given the observations. We impose a parameterized complex Gaussian prior on each column of the channel as $\mathcal{CN}(\mathbf{0}, \mathbf{\Gamma})$, where $\mathbf{\Gamma} = \mathrm{diag}(\gamma)$. Using a common hyper-parameter γ across all the columns of $\mathbf{H}^{\mathrm{v}}$ aids in promoting common row sparsity in the solution. Now, we need to obtain the posterior distribution of $\mathbf{H}^{\mathrm{v}}$, and the hyper-parameter γ. Since the prior and the noise are both Gaussian, obtaining the posterior statistics of $\mathbf{H}^{\mathrm{v}}$ is straightforward. But, computing γ requires computing the marginal probability distribution $p(\mathbf{Y}_{\mathrm{w}}; \gamma)$ and maximizing it w.r.t. γ, which is called evidence maximization or type-II maximum likelihood estimation.

To solve this, we use the Expectation Maximization (EM) algorithm, which works by lower bounding the logarithm of the evidence $p(\mathbf{Y}_{\mathrm{w}}; \gamma)$, and maximizing it iteratively. We treat $\mathbf{H}^{\mathrm{v}}$ as a hidden variable. In the expectation (E) step, we compute the expectation of the log likelihood of $(\mathbf{Y}_{\mathrm{w}}, \mathbf{H}^{\mathrm{v}})$ w.r.t. $p(\mathbf{H}^{\mathrm{v}} | \mathbf{Y}_{\mathrm{w}}, \gamma)$. In the maximization (M) step, we compute the hyper-parameter γ by maximizing the

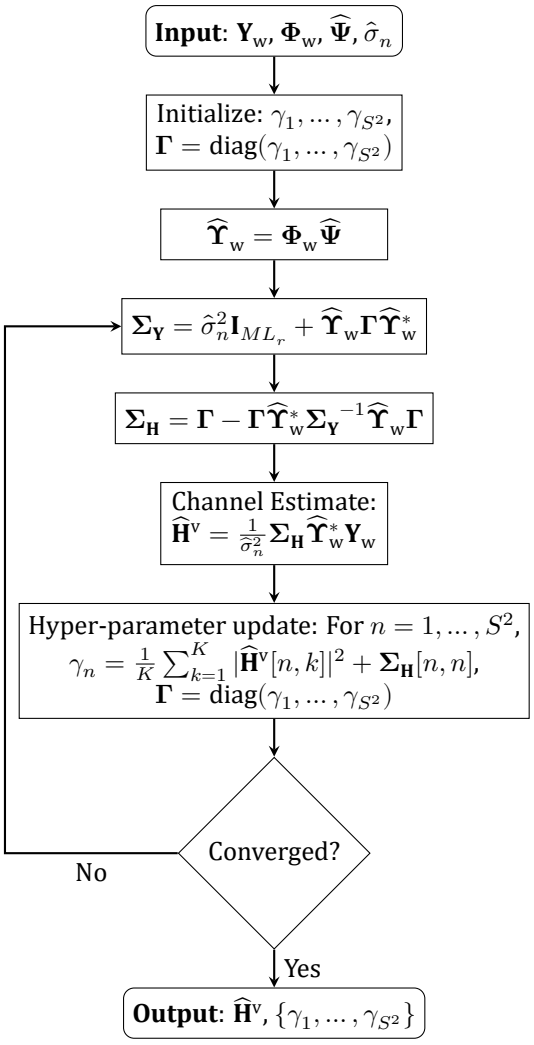


Fig. 3 – Flow diagram of MSBL.

function obtained in the E step. More details of SBL and type-II ML estimation can be found in [26, 28]. We provide a flow diagram of Multiple Measurement Vector SBL (MSBL) to compute the posterior mean and covariance of the channel, and the hyper-parameters, in Fig. 3. Specifically, in Fig. 3, the E-step of the EM algorithm corresponds to the computation of $\mathbf{\Sigma}_{\mathbf{Y}}$, $\mathbf{\Sigma}_{\mathbf{H}}$ and $\widehat{\mathbf{H}}^{\mathrm{v}}$, and the M-step corresponds to the computation of $\mathbf{\Gamma}$. We also elaborate on the E- and M-steps, albeit in the slightly different context of pattern-coupled sparse Bayesian learning, in Section 4.

Once we obtain the frequency domain channel estimate $\widehat{\mathbf{H}}^{\mathrm{v}}$, we estimate the support of the row sparse matrix and the channel coefficients using the hyper-parameters obtained using SBL. We estimate the noise variance using the Frobenius norm of the residual $\tilde{\mathbf{Y}}_{\mathrm{w}} = \mathbf{Y}_{\mathrm{w}} - \mathbf{\Phi}_{\mathrm{w}}\widehat{\mathbf{\Psi}}\widehat{\mathbf{H}}^{\mathrm{v}}$.

3.5 Denoising

By analyzing the training dataset, we observed that the channel is sparse in both the virtual beamspace and delay domains. We exploited the beamspace sparsity and obtained the frequency domain channel estimates using MLGS and SBL. In this final step, we exploit the delay domain sparsity to denoise the channel to further reduce the MSE between the original and estimated channels.

For each subcarrier k, we compute $(\widehat{\bar{\mathbf{A}}}_{\mathrm{T}} \otimes \widehat{\mathbf{A}}_{\mathrm{R}})\mathbf{H}^{\mathrm{v}}[:,k]$, and reshape it to form k^{th} subcarrier's channel matrix of size $N_r \times N_t$. Then, for each transmit and receive antenna pair, we compute a K-point inverse DFT to obtain a delay-domain channel estimate. We retain the P dominant taps in the delay-domain channel estimate, and set the other $K - P$ taps to 0. We fix P based on the estimated noise variance, and the number of training frames M. The value of P is inversely proportional to $\hat{\sigma}_n^2$, and the training dataset is used to choose an appropriate P. From our experiments on the training dataset, we found that this denoising step leads to an approximately 2 dB reduction in NMSE.

This concludes the description of the MLGS-SBL approach, and we will describe the second approach in the next section.

4. PCSBL-DDT

In this section, we present another SBL based approach to the site-specific hybrid MIMO channel estimation problem. In this method, we adapt and extend the pattern-coupled SBL in [29] to our problem, by introducing sparsity connections (or couplings) between the consecutive AoAs and AoDs. We also impose a common sparsity model on the hyper-parameters such that all the delay taps share a common support. We will show that, together, these two innovations result in accurate channel estimates.

Recall that, in *(12)*, the matrix $\mathbf{\Phi} \in \mathbb{C}^{ML_r \times N_t N_r}$ is known, and we are given the received signals $\mathbf{y}[k]$ for $k = 1, \dots, K$. We use a fixed grid, although the grid points are different for training and testing stages. Hence, the dictionary matrix $\mathbf{\Psi}$ is also known in this method.

The lag-domain representation of the channel is of length K, with $N_c \ll K$ nonzero taps, which makes the channel sparse in the time-domain. Furthermore, the nonzero taps occur in clusters. To exploit the sparsity in the time-domain, we apply the pattern-coupled SBL algorithm on the time-domain signals. As a first step, we take the inverse DFT of the received signal sequence and scale it appropriately to keep the noise variance the same, i.e.,

$$\begin{aligned}\tilde{\mathbf{y}}[d] =& \frac{1}{\sqrt{K}}\left(\sum_{k=0}^{K-1}\mathbf{y}[k]\exp\left(\frac{j2\pi kd}{K}\right)\right)\\ =& \mathbf{\Phi}\mathbf{\Psi}\tilde{\mathbf{h}}^v[d] + \tilde{\mathbf{n}}_c[d], \quad d \in \mathcal{D}, \end{aligned} \tag{22}$$

where $\tilde{\mathbf{h}}^v[d] = vec(\mathbf{\Delta}_d^v)$, and the noise $\tilde{\mathbf{n}}_c[d]$ has the same distribution as $\mathbf{n}_c[k]$. Here, $\mathcal{D} \subset \{0, \dots, K-1\}$ denotes the set of indices of the dominant delay taps. This set is determined heuristically by a simple threshold on the total energy of the received signals $\tilde{\mathbf{y}}[d]$, for $d = 0, \dots, K-1$.

This operation is done to increase the SNR by eliminating possibly all-noise samples.

As a next step, we apply a whitening filter as in the previous approach. The whitened time-domain signal is obtained similar to *(16)* as

$$\tilde{\mathbf{y}}_{\mathrm{w}}[d] = \mathbf{D}_{\mathrm{w}}^{-*}\tilde{\mathbf{y}}[d] = \mathbf{D}_{\mathrm{w}}^{-*}\mathbf{\Phi}\mathbf{\Psi}\tilde{\mathbf{h}}^{v}[d] + \tilde{\mathbf{n}}_{\mathrm{w}}[d], \qquad (23)$$

where $\tilde{\mathbf{n}}_{\mathrm{w}}[d] = \mathbf{D}_{\mathrm{w}}^{-*}\tilde{\mathbf{n}}_{c}[d] \sim \mathcal{CN}(\mathbf{0}, \sigma^2\mathbf{I}_{ML_r})$.

Note that the following approach is first applied to the true channels from the training data by regularizing it with a very small variance white Gaussian noise and uniform grids for AoAs and AoDs. Then, in the testing stage, the grid points are refined based on the joint AoA/AoD pattern that is extracted from the training data. Since the sparse model and overall procedure is the same in the training and testing phases except for the measurement matrices (there is an additional matrix $\mathbf{D}_{\mathrm{w}}^{-*}\mathbf{\Phi}$ multiplying the true channels from the left in testing stage in *(23)*), we directly present the method used in the testing stage. The channel estimator in both phases operates on the received signals $\tilde{\mathbf{y}}_{\mathrm{w}}[d]$, for $d \in \mathcal{D}$.

The pattern-coupled SBL method in [29] assumes noisy measurements of the form of

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \qquad (24)$$

where $\mathbf{y}$ is the observed vector, $\mathbf{A}$ is the measurement matrix, and the $\mathbf{x}$ is the sparse signal with some unknown block-sparsity patterns. The vector $\mathbf{n}$ is the zero-mean Gaussian noise with scaled identity covariance matrix. Hence, the model is in accordance with the one in *(23)*. Let us define $\mathbf{A} = \mathbf{D}_{\mathrm{w}}^{-*}\mathbf{\Phi}\mathbf{\Psi}$, $\mathbf{y}^d = \tilde{\mathbf{y}}_{\mathrm{w}}[d]$, $\mathbf{x}^d = \tilde{\mathbf{h}}^{v}[d]$ and $\mathbf{n}^d = \tilde{\mathbf{n}}_{\mathrm{w}}[d]$. Then, we have all the measurements from *(23)* for $d \in \mathcal{D}$ in the form

$$\mathbf{y}^d = \mathbf{A}\mathbf{x}^d + \mathbf{n}^d, \quad d \in \mathcal{D}. \qquad (25)$$

Let us express the sparse vector $\mathbf{x}^d \in \mathbb{C}^{G_t G_r}$ in the following form with special indices:

$$\mathbf{x}^d = \begin{bmatrix} x^d_{1,1} \\ \vdots \\ x^d_{G_r,1} \\ x^d_{1,2} \\ \vdots \\ x^d_{G_r,2} \\ \vdots \\ x^d_{1,G_t} \\ \vdots \\ x^d_{G_r,G_t} \end{bmatrix}, \quad d \in \mathcal{D}. \qquad (26)$$

Note that the elements of $\mathbf{n}^d$ are independent and identically distributed zero-mean complex Gaussian random variables with variance σ^2.

4.1 Proposed pattern-coupled hierarchical model

To exploit both the block-sparse structure along AoAs, AoDs, and the common sparsity for all the delay taps, we define a prior over $\mathbf{x} \triangleq \{\mathbf{x}^d : d \in \mathcal{D}\}$ as

$$p(\mathbf{x}|\boldsymbol{\alpha}) = \prod_{g_r=1}^{G_r}\prod_{g_t=1}^{G_t}\prod_{d\in\mathcal{D}} \mathcal{CN}\left(x^d_{g_r,g_t}|0, \eta^{-1}_{g_r,g_t}\right). \qquad (27)$$

To model the pattern-coupled block sparsity, we express the common parameter η_{g_r,g_t} among the delay taps as

$$\begin{aligned}\eta_{g_r,g_t} =& \alpha_{g_r,g_t} + \beta_r\alpha_{g_r-1,g_t} + \beta_r\alpha_{g_r+1,g_t} \\ &+ \beta_t\alpha_{g_r,g_t-1} + \beta_t\alpha_{g_r,g_t+1}, \end{aligned} \qquad (28)$$

where $\boldsymbol{\alpha} = \{\alpha_{g_r,g_t}\}$ are the hyper-parameters controlling the sparsity of $\mathbf{x}$. The parameters $\beta_r \in [0,1]$ and $\beta_t \in [0,1]$ indicate the pattern relevance between $x^d_{g_r,g_t}$ and its neighboring coefficients and they are taken as known constants in accordance with the related works. Different from [29], we do not impose any Gamma prior for the hyper-parameters $\{\alpha_{g_r,g_t}\}$. Instead, we consider these hyper-parameters to be deterministic and unknown, which is equivalent to assuming a non-informative prior. In our experiments, we find that this approach works better than imposing the Gamma prior.

Note that in the testing stage, the noise variance is not given explicitly. Instead a range information is provided. So, we assume that we do not know $\gamma = 1/\sigma^2$, but we introduce a uniform prior on γ, i.e., $\gamma \sim \mathcal{U}[\gamma_{\mathrm{low}}, \gamma_{\mathrm{upp}}]$ where the bounds are provided along with the test data. This assumption also differs from the Gamma distribution that is considered in [29].

We utilize an EM algorithm for learning the sparse signal $\mathbf{x}$ and the hyper-parameters $\Theta \triangleq \{\boldsymbol{\alpha}, \gamma\}$. In the EM formulation, the signal $\mathbf{x}$ is treated as a hidden variable, and we iteratively maximize a lower bound on the posterior probability $p(\Theta|\mathbf{y})$ (this lower bound is also referred to as the Q-function). The algorithm alternates between an E-step and an M-step. We explain these two steps below.

4.2 E-Step

In the E-step, we need to compute the posterior distribution of $\mathbf{x}$ conditioned on the observed data and the hyper-parameters estimated from the s^{th} iteration, i.e.,

$$p\left(\mathbf{x}|\mathbf{y}, \Theta^{(s)}\right) \propto p\left(\mathbf{x}|\boldsymbol{\alpha}^{(s)}\right) p\left(\mathbf{y}|\mathbf{x}, \gamma^{(s)}\right). \qquad (29)$$

The posterior probability can be computed as a multivariate Gaussian distribution with mean and covariance matrix for $\mathbf{x}^d$ as

$$\boldsymbol{\mu}^{d^{(s)}} = \gamma^{(s)}\left(\gamma^{(s)}\mathbf{A}^*\mathbf{A} + \mathbf{D}^{(s)}\right)^{-1}\mathbf{A}^*\mathbf{y}^d, \quad d \in \mathcal{D} \qquad (30)$$

$$\boldsymbol{\chi}^{d^{(s)}} = \left(\gamma^{(s)}\mathbf{A}^*\mathbf{A} + \mathbf{D}^{(s)}\right)^{-1}, \quad d \in \mathcal{D} \qquad (31)$$

from [29] where $\mathbf{D}^{(s)} \in \mathbb{R}^{G_r G_t \times G_r G_t}$ is a diagonal matrix with the diagonal elements $\eta^{(s)}_{g_r,g_t}$ that are ordered according to the indexing in *(26)*. Let ${\mu^d_{g_r,g_t}}^{(s)}$ and ${\chi^d_{g_r,g_t}}^{(s)}$ denote the elements of ${\boldsymbol{\mu}^d}^{(s)}$ and ${\boldsymbol{\chi}^d}^{(s)}$ corresponding to the index ordering in *(26)*.

4.3 M-Step

In the M-step, the hyper-parameters $\Theta = \{\boldsymbol{\alpha}, \gamma\}$ are estimated by treating $\mathbf{x}$ as hidden variables and iteratively maximizing the Q-function, i.e.,

$$\begin{aligned}\Theta^{(s+1)} &= \arg\max_{\Theta}\ Q\left(\Theta|\Theta^{(s)}\right)\\ &= \arg\max_{\Theta}\ \mathbb{E}_{\mathbf{x}|\mathbf{y},\Theta^{(s)}}\left[\ln p(\Theta|\mathbf{x},\mathbf{y})\right], \qquad (32)\end{aligned}$$

where the expectation is with respect to the posterior distribution $p(\mathbf{x}|\mathbf{y},\Theta^{(s)})$. We can express the above maximization with respect to Θ as

$$\begin{aligned}\underset{\Theta}{\text{maximize}}\ &\mathbb{E}_{\mathbf{x}|\mathbf{y},\Theta^{(s)}}\left[\ln p(\boldsymbol{\alpha})p(\mathbf{x}|\boldsymbol{\alpha})\right]\\ &+ \mathbb{E}_{\mathbf{x}|\mathbf{y},\Theta^{(s)}}\left[\ln p(\mathbf{y}|\mathbf{x},\gamma)p(\gamma)\right]. \qquad (33)\end{aligned}$$

We can implement the iterative updates in an alternating manner as follows:

1) Update for $\boldsymbol{\alpha}$:
Following a similar approach in [29], we can obtain a suboptimal update for $\boldsymbol{\alpha}$ as (the optimal update is not available in closed form due to the coupled variables):

$${\alpha^d_{g_r,g_t}}^{(s+1)} = \frac{|\mathcal{D}|}{\omega^{(s)}_{g_r,g_t}}, \quad d \in \mathcal{D}, \quad g_r = 1,\dots,G_r, \quad g_t = 1,\dots,G_t, \qquad (34)$$

where

$$\begin{aligned}\omega^{(s)}_{g_r,g_t} = \sum_{d\in\mathcal{D}} \Bigg(&\left|{\mu^d_{g_r,g_t}}^{(s)}\right|^2 + {\chi^d_{g_r,g_t}}^{(s)}\\ &+ \beta_r\left(\left|{\mu^d_{g_r-1,g_t}}^{(s)}\right|^2 + {\chi^d_{g_r-1,g_t}}^{(s)}\right)\\ &+ \beta_r\left(\left|{\mu^d_{g_r+1,g_t}}^{(s)}\right|^2 + {\chi^d_{g_r+1,g_t}}^{(s)}\right)\\ &+ \beta_t\left(\left|{\mu^d_{g_r,g_t-1}}^{(s)}\right|^2 + {\chi^d_{g_r,g_t-1}}^{(s)}\right)\\ &+ \beta_t\left(\left|{\mu^d_{g_r,g_t+1}}^{(s)}\right|^2 + {\chi^d_{g_r,g_t+1}}^{(s)}\right)\Bigg),\\ &g_r = 1,\dots,G_r, \quad g_t = 1,\dots,G_t. \qquad (35)\end{aligned}$$

2) Update for γ:
The hyper-parameter γ, which is the inverse of the noise variance and has a uniform prior distribution on $[\gamma_{\text{low}}, \gamma_{\text{upp}}]$ can be updated by adapting the derivation in [30] to the uniform prior considered here, as follows:

$$\gamma^{(s+1)} = \arg\max_{\gamma}\ \mathbb{E}_{\mathbf{z}|\mathbf{y},\Theta^{(s)}}\left[\ln p(\gamma)p(\mathbf{y}|\mathbf{z},\gamma)\right]. \qquad (36)$$

Algorithm 1 EM Algorithm for the Sparse Estimation of $\mathbf{x}$

Input: The set of indices of the dominating delay taps: $\mathcal{D}$. The measurement matrix: $\mathbf{A}$. The noisy measurement vectors: $\mathbf{y}^d$, $d \in \mathcal{D}$. The pattern relevance parameters: β_r and β_t. The solution accuracy: ϵ_{EM}. The minimum and maximum iteration numbers: $s_{\min}$ and $s_{\max}$. Initial hyper-parameters: $\Theta^{(0)} = \left\{\boldsymbol{\alpha}^{(0)}, \gamma^{(0)}\right\}$. The lower and upper bounds for γ: γ_{low} and γ_{upp}.
Initialize the iteration index $s \leftarrow 0$.

1: **repeat**
2: Compute $\left\{\eta^{(s)}_{g_r,g_t}\right\}$ according to *(28)* using $\boldsymbol{\alpha}^{(s)}$.
3: Update ${\boldsymbol{\mu}^d}^{(s)}$ and ${\boldsymbol{\chi}^d}^{(s)}$, for $d \in \mathcal{D}$ according to *(30)-(31)* using $\left\{\eta^{(s)}_{g_r,g_t}\right\}$ and $\gamma^{(s)}$.
4: Update $\boldsymbol{\alpha}^{(s+1)}$ and $\gamma^{(s+1)}$ according to *(34)* and *(37)* using ${\boldsymbol{\mu}^d}^{(s)}$ and ${\boldsymbol{\chi}^d}^{(s)}$, for $d \in \mathcal{D}$.
5: Set $s \leftarrow s+1$.
6: **until** $s = s_{\max}$ or $s \geq s_{\min}$ with

$$\frac{\sum_{d\in\mathcal{D}}\left\|{\boldsymbol{\mu}^d}^{(s-1)} - {\boldsymbol{\mu}^d}^{(s-2)}\right\|^2}{\sum_{d\in\mathcal{D}}\left\|{\boldsymbol{\mu}^d}^{(s-1)}\right\|^2} \leq \epsilon_{\text{EM}}. \qquad (39)$$

Output: $\hat{\mathbf{x}}^d = {\boldsymbol{\mu}^d}^{(s-1)}$, for $d \in \mathcal{D}$.

Using the uniform prior, we can obtain $\gamma^{(s+1)}$ as *(37)* at the top of the next page, where

$$\Pi_\gamma(x) = \begin{cases}\gamma_{\text{low}} & \text{if } x \leq \gamma_{\text{low}}\\ x & \text{if } \gamma_{\text{low}} < x \leq \gamma_{\text{upp}}\\ \gamma_{\text{upp}} & \text{if } x > \gamma_{\text{upp}}\end{cases}. \qquad (38)$$

The overall EM algorithm is implemented by applying the updates iteratively until the difference between ${\boldsymbol{\mu}^d}^{(s)}$ and ${\boldsymbol{\mu}^d}^{(s-1)}$ is negligible. At the final iteration, the sparse vector estimate $\hat{\mathbf{x}}^d$ is set to ${\boldsymbol{\mu}^d}^{(s)}$, for $d \in \mathcal{D}$. The overall algorithm is summarized in Algorithm 1. After multiplying $\hat{\mathbf{x}}^d$ with the dictionary matrix $\boldsymbol{\Psi}$, we obtain the time-domain channel estimates at the dominant delay taps in $\mathcal{D}$. Then, we take the K-point DFT of the time channels and scale them by $1/\sqrt{K}$ to obtain the final frequency channel estimates.

We describe the overall method in the next section in more detail.

4.4 Learning the joint relations between AoAs and AoDs

As a first step, we construct the dictionary matrix $\boldsymbol{\Psi}$ by $G_r = 96$ AoA and $G_t = 24$ AoD grid points that are uniformly selected from $[0, \pi]$. We only consider this angle range since the array steering vectors for the other angles are the same as those with the angles in $[0, \pi]$. Then using $10,000$ true frequency channels provided in the training data set, we add a white Gaussian complex noise to the time-domain channels to obtain the sparse model

$$\gamma^{(s+1)} = \Pi_{\gamma}\left(\frac{ML_r|\mathcal{D}|}{\sum_{d\in\mathcal{D}}\left(\left\|\mathbf{y}^d - \mathbf{A}\boldsymbol{\mu}^{d^{(s)}}\right\|^2 + \left(\gamma^{(s)}\right)^{-1}\left(G_rG_t - \text{tr}\left(\boldsymbol{\chi}^{d^{(s)}}\mathbf{D}^{(s)}\right)\right)\right)}\right) \quad (37)$$

$$\mathbf{y}^d_{\text{training}} = \boldsymbol{\Psi}\mathbf{x}^d_{\text{training}} + \mathbf{n}^d_{\text{training}}. \quad (40)$$

Note that the variance of the noise is selected as a very small value, e.g., 10^{-4}. The motivation is to regularize the model and apply the EM algorithm described previously without any numerical issues. We apply the EM algorithm in the previous section by keeping the inverse noise variance $\gamma = 10^4$ fixed in all the $10,000$ models obtained from the training dataset. Then, using all the sparse estimates $\hat{\mathbf{x}}^d_{\text{training}}$, we estimate the power distribution along $2G_r = 192$ AoA points and $2G_t = 48$ AoD points as in Fig. 4. Here, we apply a linear interpolation to both the AoA and AoD axes since we will utilize this in the grid construction algorithm in the testing stage. As Fig. 4 shows, some AoA/AoD grid points are more probable for the given simulation site. To exploit this learned information, we propose a grid construction algorithm, i.e., Algorithm 2, to locate the grid points more densely in the yellow regions compared to the blue regions.

We first start with a uniform grid for both AoA and AoD in $[0, \pi]$ with $96 \cdot 24$ points in total. Then, we assign additional $96 \cdot 8$ grid points to the most yellow regions in Fig. 4 by sorting the power values in decreasing order. In the next stage, we change the locations of the points to move them to the places where the power of the sparse vectors obtained from the training data is greater. At the same time, we try to prevent the neighboring grid points from being far away via judicious tuning and adjustments. For this, we consider six different distance thresholds that correspond to the maximum allowable distance between two consecutive grid points in horizontal and vertical directions. If the logarithm of the mean power value at a particular grid point is high, then a smaller (more restrictive) distance threshold is used. The motivation behind using logarithm is that the power differences across the AoA/AoD grid points are observed to be more emphasized after applying logarithm operation. In the end, the constructed grid point map is shown in Fig. 5 where the yellow points denote the selected $96 \cdot 32$ grid points to be utilized in constructing the dictionary matrix in the testing stage. Note that the minimum distance threshold value in the vector $\mathbf{d}$ is two instead of one since there is already an interpolation by a factor of two. The number of power levels, i.e., six, is chosen heuristically.

In the testing stage, after constructing the dictionary matrix $\boldsymbol{\Psi}$ according to the pattern in Fig. 5, we also modify the pattern-coupling relations accordingly. For this new grid structure, the AoA and AoD pattern-coupled block sparsity relations in *(28)* and *(35)* are modified such that

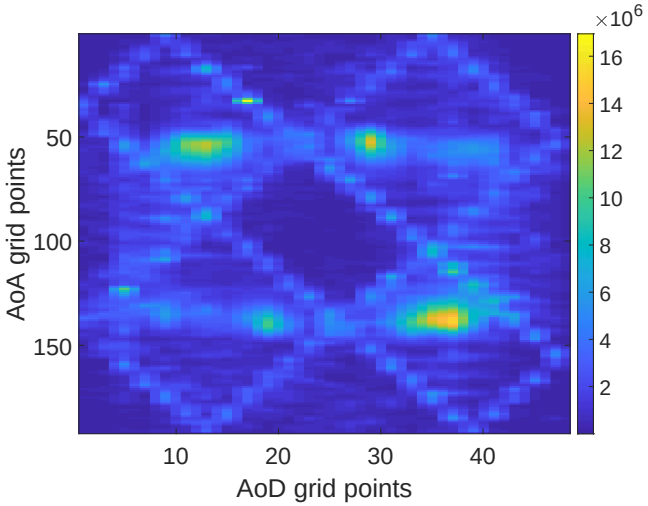


Fig. 4 – Heatmap for the power distribution of the sparse vector among AoA and AoD grid points.

the consecutive AoA and AoD gird points in Fig. 5 are constructed as coupled by keeping only the pairs with some distance threshold, i.e., not being far away more than two grid points. The updates in the EM algorithm are the same except for the indices according to the pattern-coupled block sparsity pattern.

This concludes the description of the PCSBL-DDT algorithm, and we will describe the third and last approach in the next section.

5. PC-OMP

In this section, we present the Projection Cuts Orthogonal Matching Pursuit (PC-OMP) approach for the site-specific hybrid MIMO channel estimation problem. This method makes use of the sparsity of the mm-wave channel and extracts paths parameters one by one using an OMP algorithm. A novel, custom detection method is used to detect paths, which is optimized using training data.

We express the frequency-domain channel at the k^{th} subcarrier in *(5)* as

$$\mathbf{H}[k] = \sum_{\ell=1}^{L} \tilde{\alpha}_\ell \exp\left(-j2\pi\tau_\ell k\right) \mathbf{a}_{\text{R}}(\phi_\ell)\mathbf{a}^*_{\text{T}}(\theta_\ell), \quad (41)$$

where the effect of pulse shaping and other scaling factors except the delay of the ℓ^{th} path, i.e., τ_ℓ in *(3)*, are embedded into $\tilde{\alpha}_\ell$. Using the identity $vec(\mathbf{a}\mathbf{b}^T) = \mathbf{b} \otimes \mathbf{a}$, $\mathbf{H}[k]$ vectorizes into

$$\mathbf{h}_k = \sum_{\ell=1}^{L} \tilde{\alpha}_\ell \exp\left(-j2\pi\tau_\ell k\right) \bar{\mathbf{a}}_{\text{T}}(\theta_\ell) \otimes \mathbf{a}_{\text{R}}(\phi_\ell) \quad (42)$$

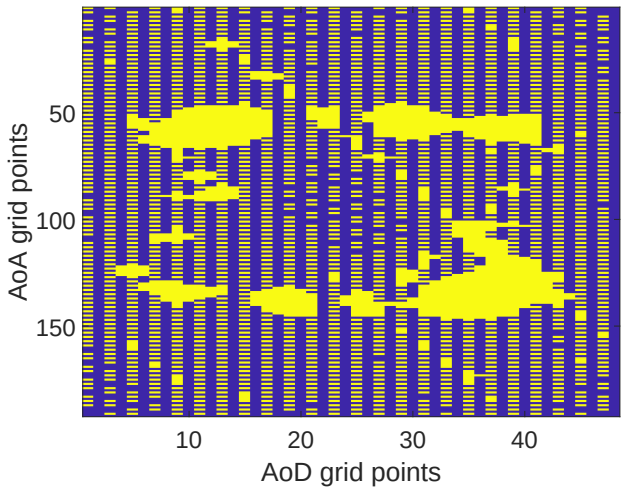


Fig. 5 – Non-uniform grid pattern for AoA and AoD in testing stage of the algorithm. The yellow pixels correspond to the selected $96 \cdot 32$ grid points.

and can be horizontally stacked into the matrix from *(13)* as

$$\hat{\mathbf{H}} = \mathbf{\Psi}\mathbf{H}^{\text{v}} = \sum_{\ell=1}^{L} \tilde{\alpha}_\ell (\bar{\mathbf{a}}_{\text{T}}(\theta_\ell) \otimes \mathbf{a}_{\text{R}}(\phi_\ell)) \mathbf{a}_{\text{F}}^T(\tau_\ell) \qquad (43)$$

with $[\mathbf{a}_{\text{F}}(\tau_\ell)]_k = \exp(-j2\pi\tau_\ell k)$. Then the stacked measurements in *(13)* can be expressed as

$$\mathbf{Y} = \mathbf{\Phi}\hat{\mathbf{H}} + \mathbf{N}_c. \qquad (44)$$

To obtain independent and identically distributed noise entries, we apply the whitening given in *(17)*, i.e.,

$$\mathbf{Y}_{\text{w}} = \mathbf{\Phi}_{\text{w}}\hat{\mathbf{H}} + \mathbf{N}_{\text{w}}. \qquad (45)$$

To ease the notation, we will define the spatial component of a path as

$$\mathbf{a}_{\text{R-T}}(\phi_\ell, \theta_\ell) = \bar{\mathbf{a}}_{\text{T}}(\theta_\ell) \otimes \mathbf{a}_{\text{R}}(\phi_\ell) \qquad (46)$$

and the channel component of a path as

$$\mathbf{a}_{\text{R-T-F}}(\phi_\ell, \theta_\ell, \tau_\ell) = \mathbf{a}_{\text{F}}(\tau_\ell) \otimes \mathbf{a}_{\text{R-T}}(\phi_\ell, \theta_\ell). \qquad (47)$$

5.1 Approach

Our strategy capitalizes on the fact that $\hat{\mathbf{H}}$ is a very sparse matrix in the sense that the number of paths L is much smaller than the maximum matrix rank $\min(N_{\text{r}}N_{\text{t}}, K)$. Hence, the path components $\mathbf{a}_{\text{F}}(\tau_\ell) \otimes \bar{\mathbf{a}}_{\text{T}}(\theta_\ell) \otimes \mathbf{a}_{\text{R}}(\phi_\ell)$ are nearly orthogonal to each other, i.e.,

$$\langle \mathbf{a}_{\text{R-T-F}}(\phi_\ell, \theta_\ell, \tau_\ell), \mathbf{a}_{\text{R-T-F}}(\phi_{\ell'}, \theta_{\ell'}, \tau_{\ell'}) \rangle \approx 0, \forall \ell \neq \ell'. \qquad (48)$$

Our algorithm consists of extracting the path parameters $(\phi_\ell, \theta_\ell, \tau_\ell)$ one by one and then subtracting their contribution in an OMP algorithm. Two key aspects here are

Algorithm 2 Grid Construction Algorithm Using the Power Distribution of the Sparse Vectors

Input: The interpolated power distribution of $\{\hat{\mathbf{x}}^d_{\text{training}}\}$ along $2G_r = 192$ AoA points and $2G_t = 48$ AoD points. Set $\mathcal{P}_{\text{min}} = 0$ and $\mathcal{P}_{\text{max}} = \infty$. Construct the vector of logarithms of the six power levels that are equally spaced between the logarithms of minimum and maximum power value in the AoA/AoD power distribution. Denote this vector by $\mathbf{p}$. Construct the corresponding distance threshold vector $\mathbf{d} = [7\ 6\ 5\ 4\ 3\ 2]^T$.

1: Select the points in the uniform grid for both AoA and AoD in $[0, \pi]$ with $96 \cdot 24$ points in total.
2: Select additional $96 \cdot 8$ grid points with the greatest power values.
3: **repeat**
4: For each grid point, compute the mean power of the three consecutive vertical and horizontal points with the considered point being at the center. Denote this mean power by p_{g_r,g_t} for grid point (g_r, g_t).
5: Count the number of entries in $\mathbf{p}$ which are less than or equal to the logarithm of p_{g_r,g_t} for each grid point (g_r, g_t). Set the corresponding distance threshold d_{g_r,g_t} as the element of $\mathbf{d}$ at this index, i.e., the number of entries.
6: Update $\mathcal{P}_{\text{max}}$ by the maximum of the mean values computed above among the non-selected grid points. Set the corresponding grid point as a candidate for inclusion.
7: Update $\mathcal{P}_{\text{min}}$ by the minimum of the mean values computed above among the selected grid points whose removal will not alter the maximum allowable distance d_{g_r,g_t} between consecutive horizontal and vertical selected points. Set the corresponding grid point as a candidate for removal.
8: Remove the grid point found in Step 7 from the grid pattern and add the grid point found in Step 6 to the grid pattern.
9: **until** $\mathcal{P}_{\text{min}} > \mathcal{P}_{\text{max}}$

Output: The updated non-uniform grid pattern

(a) how to set the dictionary for determining the path parameters and (b) how to know when to stop the OMP iterations. Key to the success of our algorithm lies in the way we address these two aspects, i.e., how we search for path parameters by using projections and how we detect the presence of a new path. We describe these in the sequel.

At each step of OMP, we want to obtain the best matching channel component, i.e., we want to solve

$$\max_{\phi_\ell, \theta_\ell, \tau_\ell} \left| \langle vec(\mathbf{\Phi}\mathbf{a}_{\text{R-T}}(\phi_\ell, \theta_\ell)\mathbf{a}_{\text{F}}^T(\tau_\ell)), vec(\mathbf{Y}_{\text{w}}) \rangle \right| \qquad (49)$$

which can be simplified into

$$\max_{\phi_\ell, \theta_\ell, \tau_\ell} |\mathbf{a}^*_{\text{R-T}}(\phi_\ell, \theta_\ell)\mathbf{\Phi}^*\mathbf{Y}_{\text{w}}\,\bar{\mathbf{a}}_{\text{F}}(\tau_\ell))|, \qquad (50)$$

where we have used $\langle vec(\mathbf{A}), vec(\mathbf{B})\rangle = tr(\mathbf{A}^*\mathbf{B})$ and the cyclic shift property of the trace.

The above maximization problem can be solved by searching over a discretized set of values of the path parameters, $(\phi_\ell, \theta_\ell, \tau_\ell)$. We start by considering a small set of values for the path parameters equally spaced in their domains. We choose a resolution of $4K$ values for τ_ℓ and of $N_\mathrm{r}/2$ ($N_\mathrm{t}/2$) values for ϕ_ℓ (θ_ℓ). Since we are considering a smaller resolution for the angles than that required for them to cover the entire angular spectrum, we substitute each $\mathbf{a}_\mathrm{R}(\phi_\ell)$ by a sector beam-pattern $\hat{\mathbf{a}}_\mathrm{R}(\phi_\ell)$ of width $4\pi/N_\mathrm{r}$. The same manipulation applies to the phase angles at the transmitter, and thus we can define $\hat{\mathbf{a}}_{\mathrm{R-T}}(\phi_\ell, \theta_\ell) = \bar{\hat{\mathbf{a}}}_\mathrm{T}(\theta_\ell) \otimes \hat{\mathbf{a}}_\mathrm{R}(\phi_\ell)$. The sector beam-pattern we consider is the same as the one defined in [31]. With this definition, we can extract a coarse version of the path parameters $(\phi_\ell, \theta_\ell, \tau_\ell)$ by maximizing

$$d_{\phi_\ell,\theta_\ell,\tau_\ell} = |\hat{\mathbf{a}}^*_{\mathrm{R-T}}(\phi_\ell, \theta_\ell)\mathbf{\Phi}^*\mathbf{Y}_\mathrm{w}\,\bar{\mathbf{a}}_\mathrm{F}(\tau_\ell)|. \qquad (51)$$

5.2 Detection

Now we want to know if those parameters can be considered as a path detection. To this end, we take into account the null hypothesis of $\mathbf{H} = \mathbf{0}$, in that case all elements of $\mathbf{Y}_\mathrm{w}$ are independent white noise and thus $\hat{\mathbf{a}}^*_{\mathrm{R-T}}(\phi_\ell, \theta_\ell)\mathbf{\Phi}^*\mathbf{Y}_\mathrm{w}\,\bar{\mathbf{a}}_\mathrm{F}(\tau_\ell)$ is also comprised of white noise components. Consequently, $d_{\phi_\ell,\theta_\ell,\tau_\ell}$ follows a Rayleigh distribution, which has the cumulative distribution funtion

$$F(d_{\phi_\ell,\theta_\ell,\tau_\ell}) = 1 - \exp\left(-\frac{d^2_{\phi_\ell,\theta_\ell,\tau_\ell}}{2\sigma^2}\right). \qquad (52)$$

Since there is a channel contribution only for a small number of path parameters, we have that the *median* of all computed values of $d_{\phi_\ell,\theta_\ell,\tau_\ell}$ should be close to that of the Rayleigh distribution $\sigma\sqrt{2\ln(2)}$. This insight is key to our algorithm. Then, σ can be approximated as

$$\sigma \simeq \mu(d_{\phi_\ell,\theta_\ell,\tau_\ell})/\sqrt{2\ln(2)}. \qquad (53)$$

In this case, the cumulative function of $\max x_k$ can be computed as $F_{\max}(x) = \prod F_k(x)$. Explicitly, it is given by

$$F_{\max}(\max(d_{\phi_\ell,\theta_\ell,\tau_\ell})) = \left(1 - \exp\left(-\frac{\max(d_{\phi_\ell,\theta_\ell,\tau_\ell})^2}{2\sigma^2}\right)\right)^{N_\mathrm{r}N_\mathrm{t}K}. \qquad (54)$$

Using this, we can compute a detection threshold corresponding to a confidence level of δ as

$$\mu(d_{\phi_\ell,\theta_\ell,\tau_\ell})\sqrt{-\log_2(1-(\delta)^{\frac{1}{N_\mathrm{r}N_\mathrm{t}K}})}. \qquad (55)$$

We compare the optimal value of $d_{\phi_\ell,\theta_\ell,\tau_\ell}$ obtained by solving *(51)* with the above threshold to decide whether the path is sufficiently significant to be included in the model, or whether to stop the OMP iterations. The value of δ is optimized using the dataset information as described later in Section 6.3.

5.3 Refinement

Once a path has been detected, we proceed to refine the path components by iterative projections. We do this by freezing two of the variables and increasing the resolution of the third, in an alternating fashion.

First steps: First, we adapt our estimation to handle a higher resolution due to the manipulation we did with the angular resolution.

We start by increasing the time resolution by computing the maximum of $|\hat{\mathbf{a}}^*_{\mathrm{R-T}}(\phi_\ell, \theta_\ell)\mathbf{\Phi}^*\mathbf{Y}_\mathrm{w}\,\bar{\mathbf{a}}_\mathrm{F}(\tau_\ell))|$ for fixed (ϕ_ℓ, θ_ℓ) and τ_ℓ with a resolution of $32K$ equally spaced points.

Then, by fixing τ_ℓ and using the identity $vec(\mathbf{A}^*\mathbf{B}\mathbf{C}) = (\bar{\mathbf{C}} \otimes \mathbf{A})^* vec(\mathbf{B})$,we can simplify the expression to

$$\left|\hat{\mathbf{a}}^*_\mathrm{R}(\phi_\ell)\overline{\mathbf{H}}(\tau_\ell)\hat{\mathbf{a}}_\mathrm{T}(\theta_\ell)\right| \qquad (56)$$

with $\overline{\mathbf{H}}(\tau_\ell)$ such that $vec(\overline{\mathbf{H}}(\tau_\ell)) = \mathbf{\Phi}^*\mathbf{Y}_\mathrm{w}\,\bar{\mathbf{a}}_\mathrm{F}(\tau_\ell)$.

We then proceed to refine the angle components with the highest number of antennas. For simplicity, let us assume that $N_\mathrm{t} > N_\mathrm{r}$. By increasing the resolution of θ_ℓ to $32N_\mathrm{t}$ equally spaced points, we do not need to use the sec-tor beam-pattern manipulation, thus we can simply maximize

$$\left|\hat{\mathbf{a}}^*_\mathrm{R}(\phi_\ell)\overline{\mathbf{H}}(\tau_\ell)\mathbf{a}_\mathrm{T}(\theta_\ell)\right| \qquad (57)$$

over θ_ℓ while the other path parameters are fixed.
Finally, we refine the expression with respect to the remaining angle. Again, the manipulation is not required, and we can maximize

$$\left|\mathbf{a}^*_\mathrm{R}(\phi_\ell)\overline{\mathbf{H}}(\tau_\ell)\mathbf{a}_\mathrm{T}(\theta_\ell)\right| \qquad (58)$$

over θ_ℓ while the other path parameters are fixed.

Iteration steps: Now that with the first steps we removed the angle uncertainty caused by the sector beam-pattern, we can proceed to repeat the same steps iteratively by substituting all sector beam-patterns $\hat{\mathbf{a}}$ by array responses $\mathbf{a}$.

Once the parameters of the path have been estimated, $(\phi_\ell, \theta_\ell, \tau_\ell)$, we use them to reconstruct the path and subtract it from the received pilots, thereby altering the residual. Then, the residual is updated, and the next path is obtained following the same steps. The residual is updated until a stopping condition is reached, as discussed in Section 5.2. How to obtain the best stopping condition for our algorithm is discussed in Section 6.3.

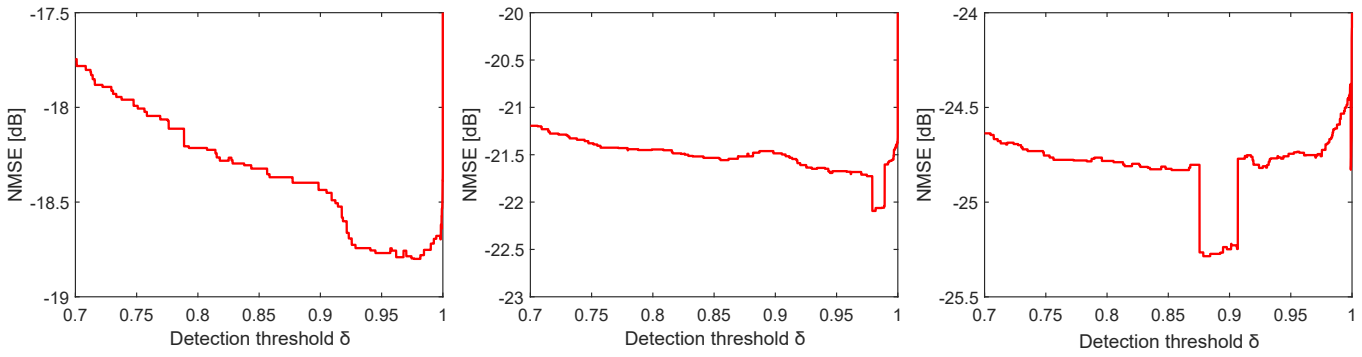


Fig. 6 – NMSE behavior over the decision threshold of the 3 training datasets.

Table 1 – NMSE table for training data

SNR (dB)	Algorithm	$-\mathbf{15}$	$-\mathbf{10}$	$-\mathbf{5}$
Pilot Frames: 20	SW-OMP	-1.45 dB	-5.70 dB	-9.68 dB
	MLGS-SBL	-4.29 dB	-9.13 dB	-12.34 dB
	PCSBL-DDT	-8.16 dB	-10.62 dB	-11.07 dB
	PC-OMP	-8.34 dB	-12.36 dB	-16.15 dB
Pilot Frames: 40	SW-OMP	-3.95 dB	-7.95 dB	-11.87 dB
	MLGS-SBL	-7.55 dB	-11.19 dB	-14.15 dB
	PCSBL-DDT	-10.56 dB	-12.14 dB	-12.62 dB
	PC-OMP	-12.66 dB	-16.33 dB	-19.78 dB
Pilot Frames: 80	SW-OMP	-7.33 dB	-11.60 dB	-15.63 dB
	MLGS-SBL	-13.02 dB	-16.37 dB	-18.94 dB
	PCSBL-DDT	-11.90 dB	-13.10 dB	-13.63 dB
	PC-OMP	-18.70 dB	-21.49 dB	-24.48 dB

Table 2 – NMSE table for test data

SNR (dB)	Algorithm	$[-\mathbf{20}, -\mathbf{11})$	$[-\mathbf{11}, -\mathbf{6})$	$[-\mathbf{6}, \mathbf{0}]$
Pilot Frames: 20	MLGS-SBL	-7.66 dB	-10.97 dB	-12.34 dB
	PCSBL-DDT	-8.94 dB	-9.99 dB	-10.31 dB
	PC-OMP	-9.09 dB	-12.45 dB	-14.22 dB
Pilot Frames: 40	MLGS-SBL	-11.87 dB	-12.79 dB	-14.20 dB
	PCSBL-DDT	-10.82 dB	-11.33 dB	-11.89 dB
	PC-OMP	-13.79 dB	-15.24 dB	-16.79 dB
Pilot Frames: 80	MLGS-SBL	-13.62 dB	-16.23 dB	-20.08 dB
	PCSBL-DDT	-11.74 dB	-12.47 dB	-12.98 dB
	PC-OMP	-16.32 dB	-19.07 dB	-23.91 dB

6. NUMERICAL RESULTS

In this section, we discuss the NMSE performance of our proposed algorithms with the training and testing data generated using *Raymobtime,* a ray tracing based mmWave channel generation tool. We train the mmWave channel estimation algorithms using $10,000$ independent channel realizations, each consisting of 100 paths between the Tx and Rx. More details about the channel generation methodology can be found in [24]. We used 20, 40, and 80 pilot frames during both the training and testing phases of the proposed algorithms. For the training phase, we used SNR values of $\{-15, -10, -5\}$ dB. We benchmark the NMSE performance of our proposed algorithms with a reference state-of-the-art model-based greedy search algorithm called SW-OMP [4].

We note that while the three new algorithms presented in this paper have been fine-tuned based on the training dataset, the baseline algorithm, SW-OMP, has been implemented as-is from the literature. On the other hand, in our implementation of SW-OMP, we consider the case where the true AoDs and AoAs are contained in the sparsifying dictionary. While the proposed algorithms do suffer from the off-grid effects, the SW-OMP algorithm is insulated from the performance degradation caused by them.

6.1 MLGS-SBL

Upon analyzing the training channels, we set the maximum number of paths obtained from MLGS to $S = 10$, and the number of levels in MLGS to $N = 5$. We set the number of columns in the initial AoD/AoA steering matrices to 256. We use the estimated noise variance after SBL to threshold the number of dominant delay taps of the channel denoiser. As this approach is primarily a model-based method, and uses few statistics from the training data, it is suitable for general mmWave channel estimation problems also. Further, the thresholds are set keeping in mind the computational complexity of the MLGS-SBL algorithm. By increasing the number of paths output by MLGS, we can potentially improve the performance of the algorithm, but at the cost of higher computational complexity. We include the NMSE values obtained for the training and testing datasets in Table 1 and Table 2, respectively. The final performance score achieved, which is a weighted combination of the NMSE performance in Table 2 when the number of pilot frames is 20, using our proposed algorithm in the mmWave channel estimation challenge is -9.16 dB.

6.2 PCSBL-DDT

In this approach, we adopted an EM-based sparse Bayesian learning method to exploit the shared sparsity between different delay taps and possible sparsity pattern couplings between consecutive AoAs and AoDs. We applied the algorithm to the time-domain received signals by only retaining the dominant delay taps to increase effective SNR in the signal used to form the channel estimate. First, we used the pattern-coupled Sparse Bayesian learning algorithm to the ground-truth channels in the training dataset by adding a small noise to regularize the data. In this way, we obtained the sparse representations for all the channels in the provided dataset. Then, using the respective sparse vectors and exploiting the density map of joint AoA/AoD grids, we selected a non-uniform grid and refined the pattern couplings between hyper-parameters. The algorithm is applied to the test dataset to obtain the channel estimates. The final performance score in the channel estimation challenge is -9.49 dB. The NMSE values for the specific scenarios are shown in Table 2 for the testing dataset.

6.3 PC-OMP

Before evaluating the performance of PC-OMP we optimize the value of the detection threshold value δ, described in Section 5.2, in order to improve the results. We create a specific optimization method for the structured problem that arises in our approach. This optimization method is focused on reducing the optimization time while being able to perform a high-resolution grid search for the parameter values. We base our training algorithm on the fact that our approach is a greedy algorithm with a carefully chosen stopping condition. This means that we can predict when a change in the solution will happen based on the selected threshold. Knowing this, we create a modified version of our approach that saves all channel iterations together with the computed threshold required for them to pass up to a minimum threshold value, in our case 0.7. Once outside the function, we can evaluate these channel estimations and compute the error as a step-wise function of the threshold. Then, we apply the average operation to the error step-wise functions for different scenarios to get a better and smoother result of the error behavior over different threshold values. Fig. 6 shows the smoothed step-wise error function for the different datasets. The selected threshold value is $\delta \approx 0.98$.

The algorithm with the custom detection method is applied to the test data and the obtained results are shown in Table 2. The PC-OMP algorithm outperforms the other two algorithms for the different data sets, as can be observed from the table. Specially, in lower SNRs where the channel estimation performance is lower due to noise, the PC-OMP algorithm achieves gains of up to 3 dB compared with the other two algorithms. At higher SNRs, e.g., $[-6, 0]$ dB, PC-OMP outperforms the other two algorithms by up to 4 dB.

The final performance score on the test dataset in the channel estimation challenge of the PC-OMP algorithm is -10.64 dB, outperforming the MLGS-SBL and the PCSBL-DDT methods by 1.48 dB and 1.15 dB, respectively. Also, from Table 1, we can see that the PCSBL-DDT and the PC-OMP algorithms are tuned better than the MLGS-SBL method for the training dataset that result in their better NMSE performances at SNR -15 dB and pilot frames $\{20, 40\}$. But the performance gap between MLGS-SBL and PCSBL-DDT reduces for the testing data for SNR $[-20, -11)$ dB and 20 pilot frames. Moreover, MLGS-SBL performs better than PCSBL-DDT at SNR $[-20, -11)$ dB and 40 pilot frames. This can be attributed to the fact that extracting more features from a training dataset may result in an excellent performance during training but slightly inferior performance while testing. This shows that a good model-based signal processing solution has to be combined with appropriate training, while taking into account the training and testing performance trade-off.

7. CONCLUSION

We have presented three novel signal processing approaches to estimate an mmWave channel in a hybrid analog-digital MIMO setup. We have adapted model-driven procedures to utilize the AoD, AoA, and channel gain information from a training dataset, and fine-tuned the algorithms to reduce the NMSE in the testing dataset. We empirically showed that our algorithms unanimously performed better than a purely model-based approach by a large margin on a given training data set. Hence, machine learning approaches can be potentially used in conjunction with model-driven based approaches to fine-tune them and thereby obtain better performance in physical layer wireless communication problems in realistic channel environments.

ACKNOWLEDGEMENTS

The work of Sai Subramanyam Thoota and Chandra R. Murthy was financially supported by research grants from Intel Inc. and the Dept. of Telecommunications, Govt. of India. The work of Ozlem Tuğfe Demir, Cenk M. Yetis, Emil Björnson, and Pontus Giselsson was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallen-berg Foundation. The work of Marios Kountouris' con-tribution was partially supported from a Huawei France-funded Chair towards Future Wireless Networks. The work of Christo Kurisummoottil Thomas was funded by a French FUI project titled MASS-START.

REFERENCES

[1] R. W. Heath, N. Gonzalez-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An overview of signal processing techniques for millimeter wave MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 3, pp. 436–453, 2016.

[2] R. Méndez-Rial, C. Rusu, N. González-Prelcic, A. Alkhateeb, and R. W. Heath, "Hybrid MIMO architectures for millimeter wave communications: Phase shifters or switches?" *IEEE Access*, vol. 4, pp. 247–267, 2016.

[3] E. Karami, "Tracking performance of least squares MIMO channel estimation algorithm," *IEEE Trans. Commun.*, vol. 55, no. 11, pp. 2201–2209, 2007.

[4] J. Rodríguez-Fernández, N. González-Prelcic, K. Venugopal, and R. W. Heath, "Frequency-domain compressive channel estimation for frequency-selective hybrid millimeter wave MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 2946–2960, 2018.

[5] A. Alkhateeb, O. El Ayach, G. Leus, and R. W. Heath, "Channel estimation and hybrid precoding for millimeter wave cellular systems," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 5, pp. 831–846, 2014.

[6] Z. Marzi, D. Ramasamy, and U. Madhow, "Compressive channel estimation and tracking for large arrays in mm-wave picocells," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 3, pp. 514–527, 2016.

[7] S. Liu, F. Yang, W. Ding, X. Wang, and J. Song, "Two-dimensional structured-compressed-sensing-based NBI cancelation exploiting spatial and temporal correlations in MIMO systems," *IEEE Trans. Veh. Tech.*, vol. 65, no. 11, pp. 9020–9028, 2016.

[8] K. Venugopal, A. Alkhateeb, N. G. Prelcic, and R. W. Heath, "Channel estimation for hybrid architecture-based wideband millimeter wave systems," *IEEE J. Sel. Areas Commun.,* vol. 35, no. 9, pp. 1996–2009, 2017.

[9] E. Vlachos, G. C. Alexandropoulos, and J. Thompson, "Wideband MIMO channel estimation for hybrid beamforming millimeter wave systems via random spatial sampling," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 1136–1150, 2019.

[10] Y. Wang, Y. Zhang, Z. Tian, G. Leus, and G. Zhang, "Super-resolution channel estimation for arbitrary arrays in hybrid millimeter-wave massive MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 947–960, 2019.

[11] F. Dong, W. Wang, Z. Huang, and P. Huang, "High-resolution angle-of-arrival and channel estimation for mmWave massive MIMO systems with lens antenna array," *IEEE Trans. Veh. Tech.*, vol. 69, no. 11, pp. 12 963–12 973, 2020.

[12] H. Xie and N. González-Prelcic, "Dictionary learning for channel estimation in hybrid frequency-selective mmWave MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7407–7422, 2020.

[13] J. P. González-Coma, J. Rodríguez-Fernández, N. González-Prelcic, L. Castedo, and R. W. Heath, "Channel estimation and hybrid precoding for frequency selective multiuser mmWave MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 2, pp. 353–367, 2018.

[14] S. Gao, X. Cheng, and L. Yang, "Estimating doubly-selective channels for hybrid mmWave massive MIMO systems: A doubly-sparse approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 5703–5715, 2020.

[15] M. Jian, F. Gao, Z. Tian, S. Jin, and S. Ma, "Angle-domain aided UL/DL channel estimation for wideband mmWave massive MIMO systems with beam squint," *IEEE Trans. Wireless Commun.*, vol. 18, no. 7, pp. 3515–3527, 2019.

[16] Z. Zhou, J. Fang, L. Yang, H. Li, Z. Chen, and S. Li, "Channel estimation for millimeter-wave multiuser MIMO systems via PARAFAC decomposition," *IEEE Trans. Wireless Commun.*, vol. 15, no. 11, pp. 7501–7516, 2016.

[17] J. Yang, C. Wen, S. Jin, and F. Gao, "Beamspace channel estimation in mmWave systems via cosparse image reconstruction technique," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4767–4782, 2018.

[18] L. Lian and V. K. N. Lau, "Coniguration optimization and channel estimation in hybrid beamforming mmWave systems with channel support side information," *IEEE Trans. Signal Process.*, vol. 68, pp. 6026–6039, 2020.

[19] C. K. Anjinappa, A. C. Gürbüz, Y. Yapıcı, and İ. Güvenç, "Off-grid aware channel and covariance estimation in mmWave networks," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3908–3921, 2020.

[20] Z. Zhou, J. Fang, L. Yang, H. Li, Z. Chen, and R. S. Blum, "Low-rank tensor decomposition-aided channel estimation for millimeter wave MIMO-OFDM systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 7, pp. 1524–1538, 2017.

[21] B. Wang, M. Jian, F. Gao, G. Y. Li, and H. Lin, "Beam squint and channel estimation for wideband mmWave massive MIMO-OFDM systems," *IEEE Trans. Signal Process.*, vol. 67, no. 23, pp. 5893–5908, 2019.

[22] F. Bellili, F. Sohrabi, and W. Yu, "Generalized approximate message passing for massive MIMO mmWave channel estimation with Laplacian prior," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3205–3219, 2019.

[23] F. Talaei and X. Dong, "Hybrid mmWave MIMO-OFDM channel estimation based on the multi-band sparse structure of channel," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1018–1030, 2019.

[24] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang, and R. W. Heath, "5G MIMO data for machine learning: Application to beam-selection using deep learning," in *Proc. IEEE Inf. Theory and App. Workshop*. IEEE, 2018, pp. 1–9.

[25] P. Schniter and A. Sayeed, "Channel estimation and precoder design for millimeter-wave communications: The sparse way," in *Proc. Asilomar Conf. on Signals, Syst., and Comput.* IEEE, 2014, pp. 273–277.

[26] D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2153–2164, 2004.

[27] Z. Zhang and B. D. Rao, "Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 5, pp. 912–926, 2011.

[28] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, no. Jun, pp. 211–244, 2001.

[29] J. Fang, L. Zhang, and H. Li, "Two-dimensional pattern-coupled sparse Bayesian learning via generalized approximate message passing," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2920–2930, 2016.

[30] J. Fang, Y. Shen, H. Li, and P. Wang, "Pattern-coupled sparse Bayesian learning for recovery of block-sparse signals," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 360–372, 2015.

[31] J. Palacios, D. De Donno, and J. Widmer, "Lightweight and effective sector beam pattern synthesis with uniform linear antenna arrays," *IEEE Antennas and Wireless Prop. Letters*, vol. 16, pp. 605–608, 2017.

AUTHORS

Sai Subramanyam Thoota (IEEE Graduate Student Member) received his B. E. degree in Electronics and Communication Engineering from the College of Engineering, Guindy, Anna University, Chennai, India, in 2003, and his M. Tech. degree in Electrical Engineering from the Indian Institute of Technology, Madras, Chennai, India, in 2006. From 2008 to 2015, he worked as a Chief engineer for Samsung Research India Bangalore, India, where he worked on GSM/GPRS/EDGE baseband transceiver design. From 2015 to 2016, he worked as a Project Associate at the Dept. of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India. Since 2016, he is pursuing his Ph. D. at the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India. His research interests include massive MIMO systems, sparse signal recovery and statistical signal processing.

Dolores Garcia Marti received her bachelor's degree in Mathematics from the University of Valencia, Spain in 2017. She received her M. Sc. in Theoretical Physics degree from Imperial College, London, UK in 2018. Since 2018, she is pursuing a Ph. D. in communications at IMDEA Networks associated to the University of Carlos III in Madrid, Spain. Her research interest lie in the intersection between machine learning and millimeter-wave communications.

Özlem Tuğfe Demir received her B. S., M. S., and Ph. D. degrees in Electrical and Electronics Engineering from Middle East Technical University, Ankara, Turkey, in 2012, 2014, and 2018, respectively. She was a Postdoctoral Researcher with Linköping University, Sweden in 2019-2020. She is currently a Postdoc with KTH Royal Institute of Technology, Sweden. She has authored the textbook *Foundations of User-Centric Cell-Free Massive MIMO* (2021). Her research interests focus on signal processing and optimization in wireless communications, massive MIMO, beyond 5G multiple antenna technologies, deep learning, and green communications.

She is a recipient of the IEEE SIU 2015 Conference Student Best Paper Award, the Best Thesis Award for M.S. Program and Graduate Courses Performance Award at the Middle East Technical University.

Rakesh Mundlamuri received his B. Tech. degree in Electronics and Communication Engineering from Shiv Nadar University, Greater Noida, India in 2018. From 2018 to 2019, he worked as a project assistant in the 5G testbed development project at the Dept. of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India. Since 2019, he is pursuing his M. Sc. in mobile computing systems at EURECOM, France. In his M. Sc. he worked on the intersection of the topics of sparse Bayesian learning and machine learning as well as real-time 5G mm-wave systems using OpenAirInterface. His research interests include machine learning in communication systems and real-time systems.

Joan Palacios received his B. Sc. degree in mathematics from the Universitat de Valencia, Spain, in 2015, and the M. Sc. degree in multimedia and communications in 2016 and PhD degree in 2020, both from IMDEA Networks and the University of Carlos III, Spain. His areas of interest lie in millimeter-wave communications with main research focus on the development of eficient schemes for beam training and tracking, hybrid analog-digital beamforming, and angle difference of arrival-based localization for quasi-optical communication systems.

Cenk M. Yetis [S'00, M'10] (cenkmyetis@ieee.org) received his B. Sc. degree in electronics engineering from Isik University, Istanbul, Turkey, in 2001, his M. Sc. degree in telecommunications engineering from Istanbul Technical University (ITU), in 2004, and his Ph. D. degree in satellite communications and remote sensing from ITU in 2010. He received The Scientiic and Technological Research Council of Turkey (TUBITAK) scholarship from 2005 to 2009. From 2003 to 2007, he was with Turk Telekom (formerly Avea), one of the top three wireless services providers in Turkey, where he held rotational responsibilities in operation and planning groups. From 2007 to 2010, he was a visiting researcher at Ohio State University and the University of California, Irvine. From 2010 to 2019, he held academic positions in universities and research centers in Hong Kong, Singapore, Turkey, and Taiwan (Province of China). In 2019, he joined Lund University, Sweden as a Researcher to extend his research and teaching activities. In 2017, he received a 2-year Academia Sinica fellowship. His research interests include signal processing, information theory, communication theory, and optimization theory for wireless communications.

Christo Kurisummoottil Thomas received his B. Tech. degree in Electronics and Com-munication Engineering from National Institute of Technology, Calicut, India, in 2010, his M. E. degree in telecommunication engineering from the Indian Institute of Science, Bengaluru, India, in 2012, and his Ph. D. degree in communication systems from EURECOM, France, in 2020. From 2012 to 2014, he was a staff design engineer with Broadcom Communication Technologies Pvt., Ltd., Bengaluru and from 2014 to 2017, he was a design engineer with Intel Corporation, in Bengaluru. Since November 2020, he has been a staff engineer with the Wireless Research and Development, Qualcomm Inc., Espoo, Finland. His research interests include areas in wireless communication, statistical signal processing, compressed sensing, and the analysis and design of mmWave systems.

Sameera Bharadwaja Hayavadana received his B. E. degree in Telecommunication Engineering from P. E. S. Institute of Technology, Bangalore, India in 2009 and his M. Tech. degree in Communication Systems from the Indian Institute of Technology Roorkee, Roorkee, India in 2011. From 2011 to 2013, he worked as a systems engineer in Innovation Labs, Tata Consultancy Services, Delhi where he worked on machine learning algorithms and applications and in 2013, he joined Samsung Semiconductor India R&D (SSIR), Bangalore, India where he is currently working as a senior staff engineer. His work at SSIR involves design and development of machine learning algorithm modules for semiconductor FAB equipment diagnosis. Since 2019, he is pursuing his Ph. D. as an external registrant from SSIR at Signal Processing for Communications lab in the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India. His research interests include group testing, sparse signal recovery and machine learning.

Emil Björnson (Senior Member, IEEE) received his M. S. degree in engineering mathematics from Lund University, Sweden, in 2007, and his Ph. D. degree in telecommunications from the KTH Royal Institute Technology, Sweden, in 2011. From 2012 to 2014, he held a joint postdoctoral position with the Alcatel-Lucent Chair on Flexible Radio,

SUPELEC, France, and the KTH Royal Institute of Technology. He joined Linköping University, Sweden, in 2014, where he is currently an associate professor. In September 2020, he became a part-time Visiting Full Professor at the KTH Royal Institute of Technology.

He has authored the textbooks *Optimal Resource Allocation in Coordinated Multi-Cell Systems* (2013), *Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency* (2017), and *Foundations of User-Centric Cell-Free Massive MIMO* (2021). He is dedicated to reproducible research and has made a large amount of simulation code publicly available. He performs research on MIMO communications, radio resource allocation, machine learning for communications, and energy eficiency . He has been on the Editorial Board of the IEEE Transactions on Communications since 2017. He has been a member of the Online Editorial Team of the IEEE Transactions on Wireless Communications since 2020. He has also been a guest editor of multiple special issues.

He has performed MIMO research for over ten years, his papers have received more than 10000 citations, and he has iled more than twenty patent applications. He is a host of the podcast Wireless Future and has a popular YouTube channel. He has received the 2014 Outstanding Young Researcher Award from IEEE ComSoc EMEA, the 2015 Ingvar Carlsson Award, the 2016 Best Ph.D. Award from EURASIP, the 2018 IEEE Marconi Prize Paper Award in Wireless Communications, the 2019 EURASIP Early Career Award, the 2019 IEEE Communications Society Fred W. Ellersick Prize, and the 2019 IEEE Signal Processing Magazine Best Column Award. He also co-authored papers that received Best Paper Awards at conferences, including WCSP 2009, the IEEE CAMSAP 2011, the IEEE SAM 2014, the IEEE WCNC 2014, the IEEE ICC 2015, and WCSP 2017.

Pontus Giselsson received his M. Sc. and Ph. D. degrees from Lund University, Sweden, in 2006 and 2012, respectively. During 2013 and 2014, he held a postdoctoral position at Stanford University. He is an associate professor in the Department of Automatic Control at Lund University. In 2012, he received the Young Author Prize at the Advanced Control of Chemical Processes IFAC Symposium; in 2014, he received the Young Author Prize at the IFAC World Congress; and in 2015, he received the Ingvar Carlsson Award from the Swedish Foundation for Strategic Research. His research interests include mathematical optimization and its wide range of applications, e.g., in machine learning, control, signal processing, and wireless communication.

Marios Kountouris (S'04–M'08–SM'15) received his diploma degree in electrical and computer engineering from NTUA, Athens, Greece in 2002 and the M. Sc. and Ph. D. degrees in electrical engineering from Télécom Paris, France in 2004 and 2008, respectively. He is currently an associate professor and Chair PI on Advanced Wireless Systems at EURECOM, France. Prior to his current appointment, he has held positions at Huawei Paris Research Center, France, CentraleSupélec, France, The University of Texas at Austin, USA, and Yonsei University, S. Korea. He was awarded a consolidator grant by the European Research Council (ERC) in 2020 on goal-oriented semantic communication. He has served as Associate Editor for the IEEE Transactions on Wireless Communications, the IEEE Transactions on Signal Processing, and the IEEE Wireless Communication Letters. He has received several awards, including the 2020 IEEE ComSoc Young Author Best Paper Award, the 2016 IEEE ComSoc CTTC Early Achievement Award, the 2013 IEEE ComSoc Outstanding Young Researcher Award for the EMEA Region, the 2012 IEEE SPS Signal Processing Magazine Award, the IEEE SPAWC 2013 Best Paper Award, and the IEEE Globecom 2009 CT Best Paper Award.

Chandra R. Murthy (S'03–M'06–SM'11) received his B. Tech. degree in Electrical Engineering from the Indian Institute of Technology, Madras in 1998, M. S. and Ph. D. degrees in Electrical and Computer Engineering from Purdue University and the University of California, San Diego, in 2000 and 2006, respectively. From 2000 to 2002, he worked as an engineer for Qualcomm Inc., where he worked on WCDMA baseband transceiver design and 802.11b baseband receivers. From Aug. 2006 to Aug. 2007, he worked as a staff engineer at Beceem Communications Inc. on advanced receiver architectures for the 802.16e Mobile WiMAX standard. In Sept. 2007, he joined the Department of Electrical Communication Engineering at the Indian Institute of Science, Bangalore, India, where he is currently working as a professor.

His research interests are in the areas of energy harvesting communications, 5G/6G technologies and compressed sensing. He has over 71 journal and 98 conference papers to his credit. He is a recipient of the MeitY Young Faculty Fellowship from the Govt. of India and the Prof. Satish Dhawan state award for engineering from the Karnataka State Government. He was an associate editor for the IEEE SIGNAL PROCESSING LETTERS during 2012-16 and the SADHANA ACADEMY PROCEEDINGS IN ENGINEERING SCIENCES during 2017-18.

He is a past Chair of the IEEE Signal Processing Society, Bangalore Chapter. He was an elected member of the IEEE SPCOM Technical Committee during 2014-19. He is currently serving as a senior area editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and as an associate editor for the IEEE TRANSACTIONS ON COMMUNICATIONS and the IEEE TRANSACTIONS ON INFORMATION THEORY.

Nuria González-Prelcic (S'96-M'00-SM'17) is currently an associate professor in the Electrical and Computer Engineering Department at North Carolina State University. Her main research interests include signal processing theory and signal processing and machine learning for wireless communications: filter banks, compressive sampling and estimation, multicarrier modulation, massive MIMO, MIMO processing for millimeter-wave communication and sensing, including vehicle-to-everything (V2X, air-to-everything (A2X and satellite MIMO communication. She has published more than 80 papers in the topic of signal processing for millimeter-wave communications. She is an Editor for the IEEE Transactions on Wireless Communications. She is a member of the IEEE Sensor Array and Multichannel Signal Processing Technical Committee. She was the founder director of the Atlantic Research Center for Information and Communication Technologies (atlanTTic at the University of Vigo from July 2008 to January 2017.

Joerg Widmer (M'06-SM'10-F'20) is Research Professor and Research Director of IMDEA Networks in Madrid, Spain. Before, he held positions at DOCOMO Euro-Labs in Munich, Germany and EPFL, Switzerland. He was a visiting researcher at the International Computer Science Institute in Berkeley, USA, University College London, UK, and TU Darmstadt, Germany. His research focuses on wireless networks, ranging from extremely high frequency millimeter-wave communication and MAC layer design to mobile network architectures. Joerg Widmer authored more than 150 conference and journal papers and three IETF RFCs, and holds 13 patents. He was awarded an ERC consolidator grant, the Friedrich Wilhelm Bessel Research Award of the Alexander von Humboldt Foundation, a Mercator Fellowship of the German Research Foundation, a Spanish Ramon y Cajal grant, as well as eight best paper awards. He is an IEEE Fellow and distinguished member of the ACM.

ENHANCED SHARED EXPERIENCES IN HETEROGENEOUS NETWORK WITH GENERATIVE AI

Neeraj Kumar[1,2], Ankur Narang[1], Brejesh Lall[2], Nitish Kumar Singh[3]
[1]Hike Private Limited, India, [2]IIT Delhi, India

NOTE: Corresponding author: Neeraj Kumar, neerajku@hike.in

Abstract – *COVID-19 has made the immersive experiences such as video conferencing, virtual reality/augmented reality, the most important modes of exchanging information. Despite much advancement in the network bandwidth and codec techniques, the current system still suffers from glitches, lags and poor video quality, especially under unreliable network conditions. In this paper, we propose the method of a video streaming pipeline to provide better video quality under erratic network conditions. We propose an environment where the participants can interact with each other through video conferencing by only sending the audio in the network. We propose a Multimodal Adaptive Normalization (MAN)-based architecture to synthesize a talking person video of arbitrary length using as input: an audio signal and a single image of a person. The architecture uses multimodal adaptive normalization, keypoint heatmap predictor, optical flow predictor and class activation map-based layers to learn movements of expressive facial components and hence generates a highly expressive talking-head video of the given person. We demonstrate the effectiveness of proposed streaming that dynamically controls the Quality of Experience (QoE) as per the requirements.*

Keywords – Audio to video generation, deep learning architecture, dynamic QoE control, GAN, multimodal adaptive normalization, video streaming pipeline

1. INTRODUCTION

The ongoing COVID-19 pandemic has forced people to work, learn, and communicate remotely on an unprecedented scale. With more people in quarantine and isolation, the demand for low latency applications, such as video streaming, online games, and teleconferencing has soared to the point that it has prompted some countries to look at ways to curb streaming data to avoid overwhelming the Internet. Several large companies have already announced that this unintended pilot on remote teleworking might become the norm.

Immersive media is likely to further exacerbate the issues related to bandwidth and latency (even in the new generation 5G networks), since all next-generation media types, either omnidirectional (360 degree) or multiview or three-dimensional, impose bandwidth requirements and latency requirements that vastly surpass those of traditional media.

With the emergence of 5G networks, ultrafast, ultra-reliable, and high bandwidth capable edge becomes an attractive option to media services developers. For immersive media, 5G is a crucial enabling technology, since its targeted key performance indicators stipulated by the architecture documents are essential to providing good Quality of Experience (QoE) for the users. With the 5G network, a videoconferencing pipeline in erratic conditions can still be challenging and advancements will be made to lower the latency and network bandwidth and provide better user experience.

A lot of work has been done on the development and optimization of novel video codecs to enhance the quality of video streaming. Various codecs have been developed to reduce the amount of streamed data while maintaining as much information as possible in the network.

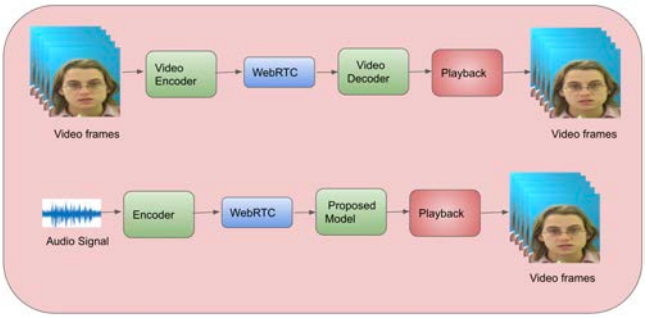


Fig. 1 – Top: Typical video streaming pipeline. In the typical system, the input video is encoded using video codecs and sent to the receiver which decodes it in the form of a lossy reconstruction that preserves most of the video features at a pixel level. Bottom : Proposed streaming pipeline where the audio signal is sent through a general-purpose WebRTC DataChannel and at the receiver side, the proposed model converts the audio into the video signal.

In a typical system (Fig. 1), the data is first read from a video source and compressed. The compressed data is sent over a network to the receiving end, where a decoding algorithm reconstructs a representation of the original feed from the streamed data. Since most of the codecs are lossy, the reconstruction process at the receiver end does not create the original feed but sufficiently close to the original with some distortions. The compression techniques utilize the fact that not all the information contained within a video frame is

equally important and prioritize the preservation of more important aspects of a feed over others in the compression/decompression process. Despite these advancements, a lot of work needs to be done in order to give an enhanced videoconferencing experience in unreliable network conditions [1] such as glitches, lags, low internet bandwidth, etc.

In this paper, we propose the audio driven video conferencing methodology that helps in improving the video quality in odd network scenarios. In the proposed method, we have used a GAN-based approach at the receiver's end to generate video with enhanced quality under unreliable conditions. One of the possible concerns of this methodology is that it shifts the burden from communication bandwidth to increased computation at the receiver end. The use of a GAN-based [2] approach can increase the latency resulting in the lag of video during streaming. But with the rapid improvement of hardware capabilities in mobiles and personal computers, this is unlikely to be a major obstacle. With the recent development of NVIDIA Maxine project [3], such hurdles can be resolved and results into the practical system that provides immense gains over the conventional methods.

Given an arbitrary image and an audio sample, we propose multimodal adaptive normalization in the proposed architecture to generate realistic videos. We have built the architecture based on [4] to show how multimodal adaptive normalization helps in generating highly expressive videos using the audio and person's image as input. The proposed GAN architecture consists of generator and discriminator. The generator has two major components, namely multimodal adaptive normalization framework and class activation attention map. A multimodal adaptive normalization framework feeds various features such as optical flow/keypoint heatmaps, single image, audio melspectrogram, pitch and energy of the audio frames to the generator to produce realistic and expressive video. A class activation attention map helps the generator to produce global features such as eyes, nose, lips, etc and local features such as movements of facial action units properly which will increase the video quality. The discriminator used in the proposed method is multiscale with a class activation attention layer to discriminate fake and real frames at the global and local level.
Our main contributions are :

- The proposed speech driven facial video synthesis architecture is a GAN-based approach that consists of a generator and discriminator in Section 4. The generator incorporates the multimodal adaptive normalization framework (Fig. 9), optical flow/keypoint predictor and class activation map-based attention layer to generate the expressive videos. The discriminator uses multiscale patchGAN-based discriminator along with a class activation map-based layer to classify fake or real images.

- We have shown how the Quality of Experience (QoE) in videoconferencing has improved in low bandwidth networks by the proposed architecture in Section 7.2.2. The proposed videoconferencing pipeline helps in controlling the QoE based on the compute resource, bandwidth availability and importance of the speaker in the videoconference. It can further be used in data privacy by synthesizing the video on person or avatar. Noisy audio can be handled by the proposed model and generates the expressive output and gives a high quality of experience.

- Various experimental (Section 7.2) and ablation studies (Section 7.3) have shown that the proposed multimodal adaptive normalization is flexible in building the architecture with various networks such as 2DConvolution, partial2D convolution, attention, LSTM, Conv1D for extracting and modeling the mutual information.

- The proposed multimodal adaptive normalization-based architecture for video synthesis using audio and a single image as an input has shown superior performance on multiple qualitative and quantitative metrics such as Structural Similarity Index (SSIM), Peak Signal to Noise Ratio (PSNR), Cumulative Probability of Blur Detection (CPBD), Word Error Rate (WER), blinks/sec and Landmark Distance (LMD) in tables 1, 2, 3 and 4. The generated videos are given at [1].

2. BACKGROUND

2.1 Audio to video generation

Audio to video generation is an active area of research due to its wide range of applications such as for the entertainment industry, education, healthcare and many more. Computer Generated Imagery (CGI) has become an important part of the entertainment industry due to its ability to produce high quality results in a controllable manner.

Facial animation is an important part of CGI as it is capable of conveying a lot of information not only about the character but also about the scene in general. The generation of realistic and expressive animation is highly complex due to its multiple properties such as lip synchronization with audio, movements of a facial action units for expressiveness and natural eye blinks. Facial synthesis in CGI is traditionally performed using face capture methods, which have seen drastic improvements over the past years and can produce faces that exhibit a high level of realism. However, these approaches require expensive equipment and significant amounts of labour. In order to drive down the cost and time required to produce high quality, researchers are looking into automatic

[1] https://sites.google.com/view/itu2021

face synthesis using machine learning techniques.

Machine learning methods could simplify the video generation process by automatically producing it from the audio. Such methods could be applied in post-production of film making to achieve better lip synchronization. They can be applied in the education sector to teach students in a more realistic manner that can help reduce the cost of teaching. Apart from that, such techniques can be used to generate parts of the face that are occluded or missing in a scene. This technology can improve band-limited visual telecommunications by either generating the entire visual content based on the audio or filling in dropped frames.

2.2 Facial video

Facial video generation is a complex problem. It has several properties which make the video realistic.

- Semantic consistency - The facial features such as eyes, nose, lips, etc. should be consistent among each other.
- Temporal consistency - Video consists of several frames. Each frame should be temporal smoother with its previous and next frames, so that there should not be any jitters, spikes or holes in the video.
- Expressiveness - This property makes the video more realistic and natural. Properties such as movement of facial action units, lip synchronization with the audio and the blinking of eyes make the video more realistic and visually appealing.

While generating the video from audio, the predicted videos should inhibit such properties. Optical flow and a keypoint heatmap help in making the video semantically and temporally consistent as well as more expressive.

2.2.1 Optical flow

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is a 2D vector field where each vector is a displacement vector showing the movement of points from the first frame to the second. Optical flow has many applications in areas such as structure from motion [5], video compression [6] and video generation. The optical flow helps in achieving the temporally smoother videos. Fig. 2 shows the optical flows between the two consecutive frames of any videos. The optical flow gives the temporal as well as spatial information based on the movement of the intensity values of the frames.

2.2.2 Keypoint heatmap

Facial landmark detection is a well-studied topic in the field of computer vision with many applications such as face verification [7], face recognition [8], and facial attribute inference [9]. The high variability of shapes, poses, lighting conditions, and possible occlusions makes it a particularly challenging task even today. Such variabilities can be captures using the facial landmark keypoints. We detect the landmark keypoints around the cheeks, nose, eyes, lips to capture the movement of face while speaking or giving expressions using deep learning techniques. The heatmap of keypoints helps in giving a coarser view of these keypoint locations. Such heatmaps help the model to focus on the regions around the lips, noise, eyes and cheeks such that it captures the expressiveness of the image. Fig. 3 shows the landmark points of the images on the upper part of the image. The lower part shows the heatmap of the keypoints which gives the information about the expressiveness of the images.

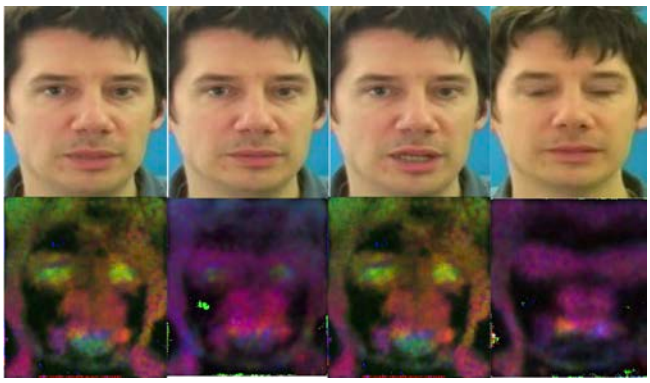


Fig. 2 – Top: Frames of the video. Bottom: Optical flow of the video.

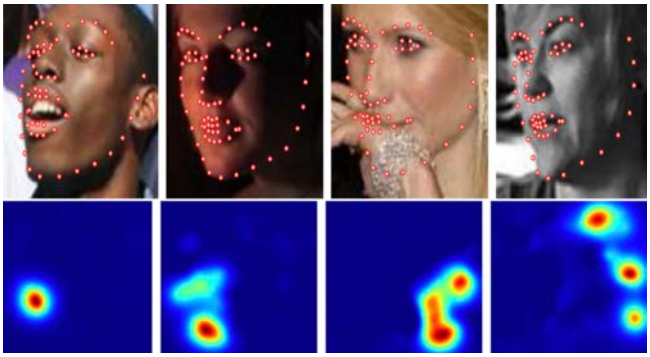


Fig. 3 – Top: facial keypoints. Bottom: keypoint heatmap of the face.

2.3 Audio

Modeling audio is a complex problem. Several aspects of the synthesized speech, such as a speaker's voice, speaking style/prosody and noise comes into play to better incorporate the audio into modeling. The range of prosody in the dialogue must encompass a large range of human conversation, from neutral expression to extremely emotional, while always sounding perfectly natural. Here, prosody refers to the variation of several speech related phenomena such as intonation, stress, rhythm and style of the speech. Traditionally, prosody modeling is based on schematizing and labeling prosodic phenomena and developing rule-based systems or statistical models from

the derived data. However, the prosodic attributes are difficult and time consuming to annotate. The prosody of speech is best captured by pitch, energy and melspectrogram of the audio frames. Such features help the deep learning model to incorporate natural and expressive audio to meet the end tasks such as generation of expressive video.

2.3.1 Pitch

Pitch is the fundamental frequency of an audio waveform, and is an important parameter in the analysis and synthesis of speech and music. Normally only voiced speech and harmonic music have well-defined pitch. But we can still use pitch as a low-level feature to characterize the fundamental frequency of any audio waveform. The typical pitch frequency for human speech is between 50 and 450 Hz, whereas the pitch range for music is much wider.

2.3.2 Energy

Energy models the excitation pattern on the basilar membrane by simulating the acoustic signal transformations in the ear according to the perceptual model of the human auditory system. Short-term speech energy is closely related with activation or arousal dimension of the emotion, its usage in the conventional features contributes to the classification of emotions.

2.3.3 Melspectrogram

A melspectrogram is a spectrogram where the frequencies are converted to the mel scale.This mel scale is constructed such that sounds of equal distance from each other on the mel scale, also "sound" to humans as they are equal in distance from one another. In contrast to the Hz scale, where the difference between 500 and 1000 Hz is obvious, whereas the difference between 7500 and 8000 Hz is barely noticeable.

2.4 Generative adversarial network

The Generative Adversarial Network (GAN) [10] consists of the generative model and discriminative model. The GAN framework naturally takes up a game-theoretic approach. The word "adversarial" is chosen as the two networks, i.e., generator and discriminator are in constant conflict and compete with each other. The generative model can be thought of as analogous to a team of counterfeiters, trying to create money similar to the real ones while the discriminator acts as police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods by constantly giving knowledge and feedback until the counterfeits are indistinguishable from the genuine articles.

The generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. We can train both models using only the highly successful back propagation and dropout algorithms and sample from the generative model using only forward propagation.

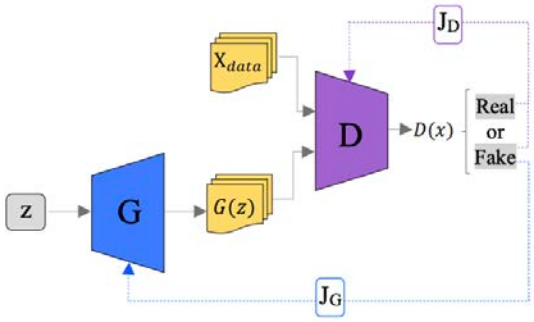


Fig. 4 – Architecture of Generative Adversarial Network (GAN)

Fig. 4 shows the general architecture of GAN. To learn the generator's distribution p_g over data x, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z;\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(x;\theta_d)$ that outputs a single scalar. $D(x)$ represents the probability that x came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1-D(G(z)))$: In other words, D and G play the following two-player minimax game with value function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))].$$

2.5 Normalization

The normalization framework has become the integral part of neural network training. It has gained success due to many reason such as a higher learning rate, faster training, regularization effects, smoothing of loss landscape, etc. The variants of normalization are discussed in the following subsections. One of the first normalization architecture proposed was batch normalization [11] which helps the deep learning community understand the effect of normalization.

2.5.1 Batch normalization

In traditional deep networks, a too-high learning rate may result in the gradients that explode or vanish, as well as getting stuck in poor local minima. Batch normalization [11] helps address such issues. By normalizing activations throughout the network, it prevents small changes to the parameters from amplifying into larger and suboptimal changes in activations in gradients; for instance, it prevents the training from getting stuck in the saturated regimes of nonlinearities.

The Dropout [12] is typically used to reduce overfitting but in a batch-normalized network it can be either removed or reduced in strength and helps in better generalization of the network. Batch normalization reduces the photometric distortions because batch normalized networks train faster and observe each training example fewer times, we let the trainer focus on more "real" images by distorting them less.

Equation *(4)* is the batch normalized output with input($x_1 \cdots x_n$) used to calculate the mean (Equation *(1)*) and variance (Equation *(2)*) which is used to get the normalized output ($\hat{x}_1 \cdots \hat{x}_n$) (Equation *(3)*). Need of normalization occurs as distribution invariance assumption is not satisfied at local level. Without normalization, the model has to run more steps for parameters to adapt. Use of scale (γ) and bias (β) in Equation *(4)* gives flexibility to work with normalized input and also with scaled normalized input, if there is a need, thus increasing the representation power.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad (1)$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \qquad (2)$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \qquad (3)$$

$$y_i = \gamma \hat{x}_i + \beta \qquad (4)$$

2.5.2 Variants of normalization

Variants of normalization have been used to capture various information such as style, texture, shape, etc. Instance Normalization (IN) [13] is a representative approach which was introduced to discard instance-specific contrast information from an image during style transfer. Inspired by this, adaptive instance normalization [14] provided a rational interpretation that IN performs a form of style normalization, showing that by simply adjusting the feature statistics, namely the mean and variance of a generator network, one can control the style of the generated image. IN dilutes the information carried by the global statistics of feature responses while leaving their spatial configuration only, which can be undesirable depending on the task at hand and the information encoded by a feature map. To handle this, Batch-Instance Normalization(BIN) [15] normalizes the styles adaptively to the task and selectively to individual feature maps. It learns to control how much of the style information is propagated through each channel of features leveraging a learnable gate parameter. For style transfer across the domain, UGATIT [16] has used adaptive instance and Layer Normalization (LN) [17] which adjusts the ratio of IN and LN to control the amount the style transfers from one domain to other domains.

For style transfer tasks, a popular methodology is trying the denormalization to the learned affine transformation that is parameterized based on a separate input image (the style image). SPADE [18] makes this denormalization spatially sensitive. SPADE normalization boils down to "conditional batch normalization which varies on a per-pixel basis". In world-consistent video to video synthesis [19], they have used optical features and semantic maps in the normalization to learn the affine parameters to generate the realistic and temporally smoother videos.

We have proposed multimodal adaptive normalization to incorporate the higher-order statistics of multimodal fea- tures (image and audio) through affine parameters of nor- malization i.e. scale (γ) and shift (β) .

3. RELATED WORK

There have been many years of research on video codecs for various applications such as AV1 [20] and VVC [21] codecs. Researchers are working on improving the codes using machine learning techniques either by end to end approaches or working on specific parts of video streaming pipelines.

In one of the approaches , face detection/mesh extraction [22, 23, 24, 25] and on body pose tracking [26, 27, 28], focusing on both 3D and 2D meshes, generally based on neural networks are used to encode the video streams and sent to the data channel. The final video is then reconstructed back by using body pose along with mesh at the receiver side to make the video streaming pipelines in erratic network conditions.

There was some work on video compression and reconstruction based on facial landmarks in [29, 30], which are promising in extremely low bitrates, but did not demonstrate real-time conferencing capabilities.

3.1 Audio to realistic video generation

The earliest methods for generating videos relied on Hidden Markov Models which captured the dynamics of audio and video sequences. Simons and Cox [31] used the Viterbi algorithm to calculate the most likely sequence of the mouth shape given particular utterances. Such methods are not capable of generating quality videos and lack emotions.

3.1.1 Phoneme and visemes generation of videos

Phoneme and viseme-based approaches have been used to generate videos. Real-Time Lip Sync for Live 2D Animation [32] has used an LSTM-based approach to generate live lip synchronization on 2D character animation.

Some of these methods target rigged 3D characters or meshes with predefined mouth blend shapes that correspond to speech sounds [33, 34, 35, 36, 37, 38] which have primarily focused on mouth motions only and show a finite number of emotions, blinks, facial action units movements.

3.1.2 Deep learning techniques for video generation

CNN-based architectures for audio to video generation: A lot of work has been done on CNN to generate realistic videos given an audio and static image as input. [39](Speech2Vid) has used encoder-decoder architecture to generate realistic videos. They have used L1 loss between the synthesized image and the target image. Our approach has used multimodal adaptive normalization in GAN-based architecture to generate realistic videos.

Synthesizing Obama: Learning lip sync from audio [38] is able to generate quality videos of Obama speaking with accurate lip-sync using RNN-based architecture. They can generate only a single person video whereas the proposed model can generate videos on multiple images in GAN-based approach.

GAN-based architectures for audio to video generation: Temporal Gan [40] and generating videos with scene dynamics [41] have done the straightforward adaptation of GANs for generating videos by replacing 2D convolution layers with 3D convolution layers. Such methods are able to capture temporal dependencies but require constant length videos. The proposed model is able to generate videos of variable length with a low word error rate.

Realistic Speech-Driven Facial Animation with GANs (RSDGAN) [42] used a GAN-based approach to produce quality videos. They used identity encoder, context encoder and frame decoder to generate images and used various discriminators to take care of different aspects of video generation. The proposed method has used multimodal adaptive normalization along with class activation layers and an optical flow predictor and keypoint heatmap predictor in the GAN-based setting to generate expressive videos.

The X2face [43] model uses a GAN-based approach to generate videos given a driving audio or driving video and a source image as input. The model learns the face embeddings of source frame and driving vectors of driving frames or audio bases which generate the videos. In X2face, the video is processed at 1fps whereas the model generate the video at 25fps. The quality of output video is not good as compared to our proposed method with audio as an input.

MoCoGAN [44] uses RNN-based generator with separate latent spaces for motion and content. A sliding window approach is used so that the discriminator can handle variable-length sequences. This model is trained to generate disentangled content and motion vectors such that they can generate audios with different emotions and content. Our approach uses multimodal adaptive normalization to generate expressive videos.
[45] extracts the expression and pose from an audio signal and a 3D face is reconstructed on the target image. The model renders the 3D facial animation into video frames using the texture and lighting information obtained from the input video. Then they fine-tune these synthesized frames into realistic frames using a novel memory-augmented GAN module. The proposed approach uses multimodal adaptive normalization with predicted optical flow/keypoint heatmap as an input to learn the movements and facial expressions on the target image with audio as an input. CascadedGAN [46] have used the L-GAN and T-GAN for motion (landmark) and texture generation. They have used a noise vector for blink generation. Model Agnostic Meta Learning (MAML) [47] is used to generate the videos on an unseen person image. The proposed method has used multimodal adaptive normalization to generate realistic videos.

[48] uses an Audio Transformation network (AT-net) for audio to landmark generation and a visual generation network for facial generation. [49] uses audio, identity encoder and a three-stream GAN discriminator for audio, visual and optical flow to generate lip movement based on input speech. [50] enables arbitrary-subject talking face generation by learning disentangled audiovisual representation through an associative-and-adversarial training process. [51] uses a generator that contains three blocks: (i) Identity Encoder, (ii) Speech Encoder, and (iii) Face Decoder. It is trained adversarially with a visual quality discriminator and pretrained architecture for lip audio synchronization. [49, 50, 51] are limited to lip movements whereas the proposed method uses multimodal adaptive normalization to generate different facial action units of an expressive video. [52] uses Asymmetric Mutual Information Estimator (AMIE) to better express the audio information into generated video in talking face generation. They have AIME to capture mutual information to learn the cross-modal coherence whereas we have used multimodal adaptive normalization to incorporate multimodal features into our architecture to generate the expressive videos. [4] have used deep speech features into the generator architecture with spatially adaptive normalization layers in it along with lip frame discriminator, temporal discriminator and synchronization discriminator to generate realistic videos. They have limited blinks and lip synchronization whereas the proposed method used multimodal adaptive normalization to capture the mutual relation between audio and video to generate expressive video.

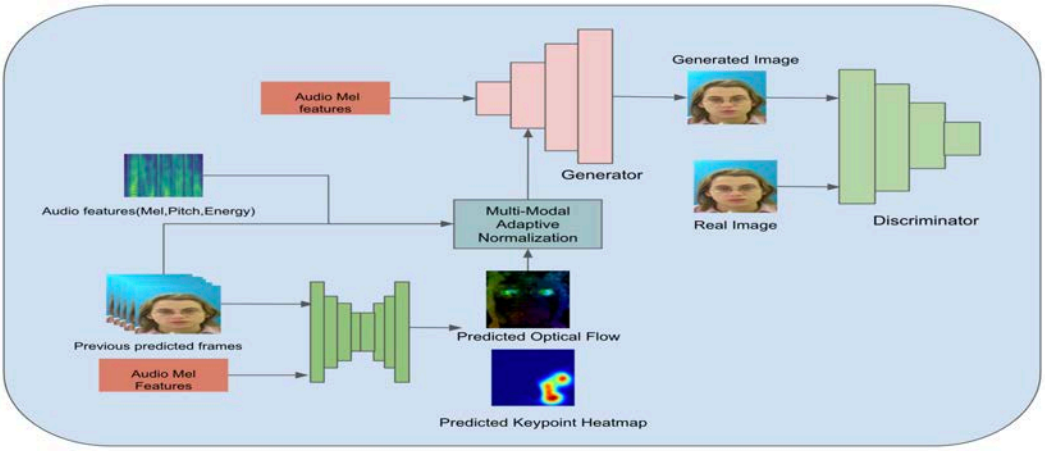


Fig. 5 – Proposed architecture for audio to video synthesis

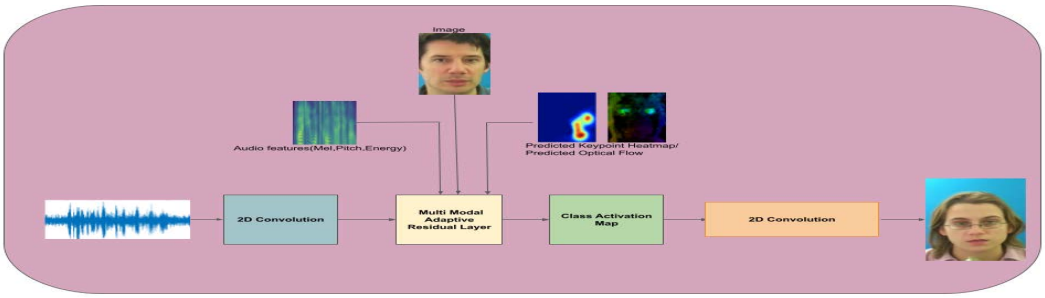


Fig. 6 – Generator architecture

4. ARCHITECTURAL DESIGN OF SPEECH DRIVEN VIDEO SYNTHESIS

Given an arbitrary image and an audio sample, the proposed method is able to generate speech synchronized realistic video on the target face. The proposed method uses multimodal adaptive normalization technique to generate realistic expressive videos. The proposed architecture is GAN-based which consists of a generator and a discriminator; see Fig. 5.

The architecture consists of 4 important subparts i.e. Generator, Discriminator, Multimodal Adaptive Normalization and Features Extractor Modules. The role of the generator is to generate realistic video frames (Fig. 6). The discriminator distinguishes between real and fake images and helps the generator to produce more realistic images (Fig. 14). The multimodal adaptive normalization provides necessary information/features i.e. pitch, energy and Audio Melspectrogram Features (AMF) from audio domain & static image and Optical Flow (OF)/facial Keypoint Heatmap (KH) features from video domain to the generator (figures 7, 10, 11). The feature extractor modules consists of various predictor modules such optical flow predictor, keypoint heatmap predictors, pitch, energy and audio melspectrogram extractors that extract necessary features such as Optical Flow (OF)/facial Keypoints Heatmap (KH), pitch, energy and melspectrogram features which go into the normalization framework.

4.1 Generator

Fig. 6 shows the generator architecture to generate realistic images. It consists of convolution layers, several layers having multimodal adaptive normalization-based Resnet [53] block (MANResnet) along with a class activation map layer. Fig. 7 shows the residual architecture around Multimodal Adaptive Normalization (MAN) along with 2d convolution and Relu [54] activation layers. The audio and video features namely person's image, predicted optical flow/predicted keypoint heatmap, melspectrogram features, pitch and energy go into the multimodal adaptive normalization network. Figures 10 and 11 show the multimodal adaptive normalization architecture which takes various features of audio and video domain and calculates the affine parameters i.e, scale, γ and a shift, β for normalization.

Class Activation Map (CAM)-based layer: This layer is employed as a layer of generator to capture the global and local features of the face. In class activation map [55], the concatenation of adaptive average pooling and adaptive max pooling of feature map create the CAM features which capture global and local facial features respectively. It helps the generator to focus on the image regions that

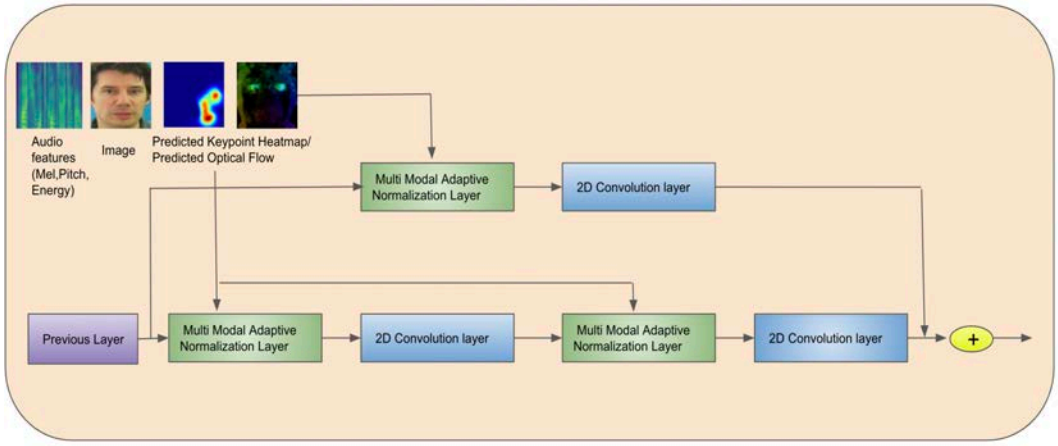


Fig. 7 – Multimodal adaptive normalization residual architecture

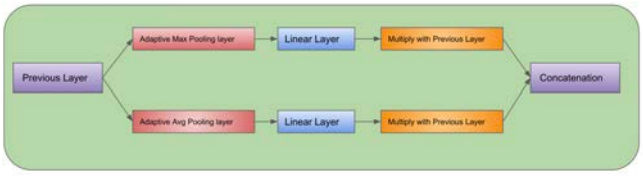


Fig. 8 – Class activation map layer architecture in generator

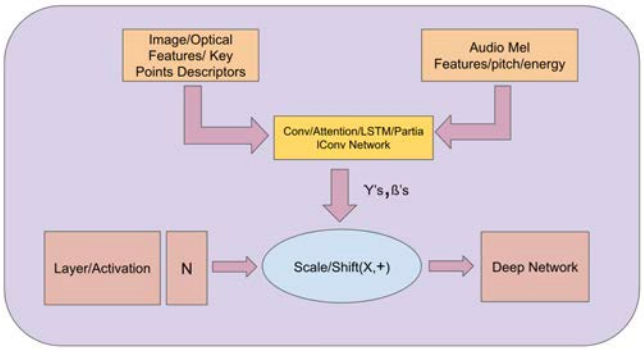


Fig. 9 – Higher level architecture of multimodal adaptive normalization

are more discriminative such as eyes, mouth and cheeks (Fig. 8).

4.2 Multimodal adaptive normalization

Fig. 9 shows the higher-level architectural design of multimodal adaptive normalization. The affine parameters i.e, scale, γ and a shift, β are typically used to learn the higher-order statistics of image features corresponding to style, texture, etc. to generate the required output as depicted in various previous work [13, 56, 18, 16, 19, 15]. We are the first ones to propose how affine parameters help to learn the higher-order statistics of multiple domains. The respective affine parameters i.e. γ and β are dynamically controlled by learnable parameters, ρ' s whose sum will be 1 constrained by the softmax function (Equation *(6)*). The idea behind using multimodal adaptive normalization is that various features in the multimodal domain are correlated. Multimodal adaptive normalization opens the non-trivial path to capture the mutual dependence between various domains. Generally, various encoder architectures [42] are used to convert the various features of multiple domains into latent vectors, and then the concatenated vectors are fed to the decoder to model the mutual dependence and generate the required output. The proposed multimodal adaptive normalization helps in reducing the number of model parameters required to incorporate multimodal mutual dependence into the architecture.

In the multimodal adaptive normalization where we have used the pitch, energy and Audio Melspectrogram Features (AMF) (Figure 11) from audio domain & static image and Optical Flow (OF)/facial Keypoints Heatmap (KH) features from video domain (Figure 10) in the normalization to compute the different affine parameters in multimodal adaptive normalization setup. Multimodal adaptive normalization gives the flexibility of using various architectures namely 2D convolution, partial convolution and attention model for video related features and 1D convolution and the LSTM layer for audio features, as shown in Table 6.

(Equation *(5)*) shows the combined equation of the multimodal adaptive normalized output where x_{IN} is the instance normalized with mean and variance calculated across batch and channel.Various γ's and β's are modeled and linearly combined under an equation. The parameter ρ's is used to combine these parameters (Equation *(6)*). The value of ρ's is constrained to the range of [0, 1] by using the softmax function (Equation *(6)*).

$$y = \rho_1(\gamma_{Image}x_{IN} + \beta_{Image}) + \rho_2(\gamma_{OF/KH}x_{IN} + \beta_{OF/KH}) + \rho_3(\gamma_{AMF}x_{IN} + \beta_{AMF}) + \rho_4(\gamma_{pitch}x_{IN} + \beta_{pitch}) + \rho_5(\gamma_{energy}x_{IN} + \beta_{energy}) \quad (5)$$

$$\rho_1 + \rho_2 + \rho_3 + \rho_4 + \rho_5 = 1 \quad (6)$$

4.3 Feature extractor modules

This section consists of various feature extractor modules which extract the various features such as pitch, energy

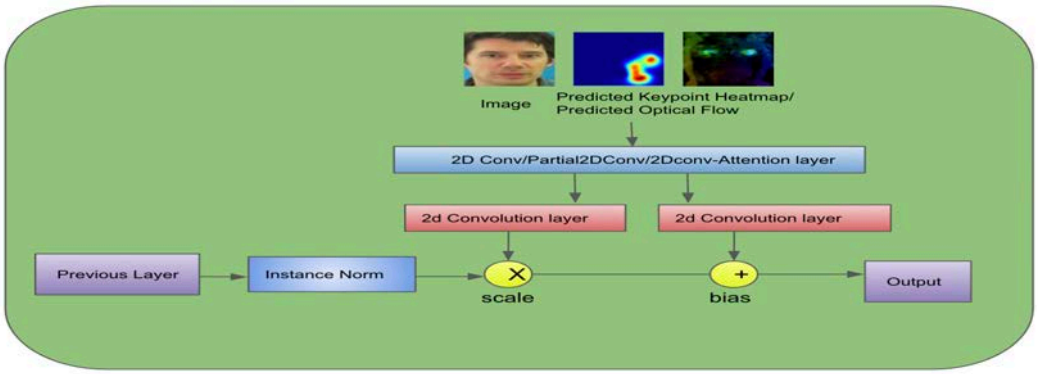


Fig. 10 – Architecture to calculate the affine parameters from video features in multimodal adaptive normalization

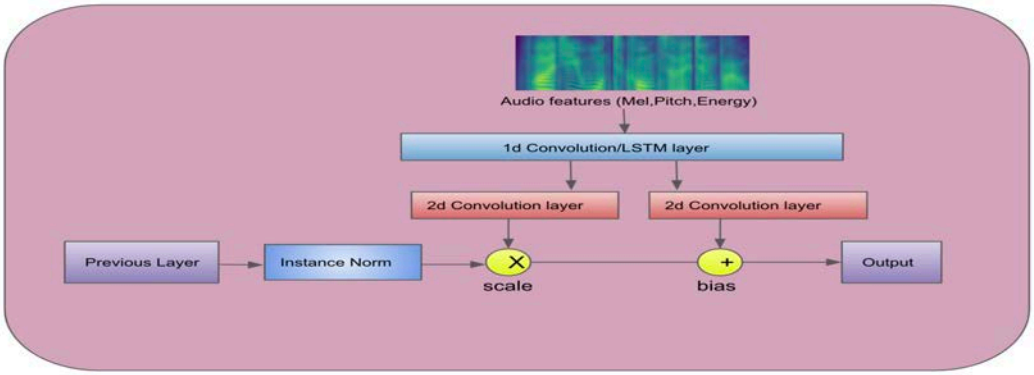


Fig. 11 – Architecture to calculate the affine parameters from audio features in multimodal adaptive normalization

and Audio Melspectrogram Features (AMF) from audio domain & static image and Optical Flow (OF)/facial Keypoints Heatmap (KH) features from video domain. There are various feature extractor modules such as keypoint heatmap predictor which predicts the keypoint heatmap having the information of various facial parts, e.g., upper left eyelid and nose bridge. Another module is Optical Flow Predictor which predicts the optical flow of the frame needed to maintain the temporal consistency in any video.

Fig. 7 shows that block level diagram of how the various features of the audio and video domain go into the multimodal adaptive normalisation through the affine parameters i.e, scale, γ and a shift, β .Fig. 10 shows that the video features such as predicted optical flow or keypoint heatmaps and single image go into a few layers of 2D convolution/2D partial convolution/2D convolution-attention layers to generate the corresponding γ's and β's and then go to the normalization layers of the generator. Fig. 11 shows that various features such as pitch, energy and melspectrogram go to the various layers of 1D convolution/LSTM to generate the corresponding γ's and β's.

4.3.1 Keypoint heatmap predictor

The predictor model is based on Hourglass architecture [57] that, from the input image, estimates K heatmaps $H_K \in [0, 1]^{H \times W}$, one for each keypoint, each of which represents the contour of a specific facial part, e.g., upper left eyelid and nose bridge. It captures the spatial configuration of all landmarks, and hence it captures pose, expression and shape information. We have used a pretrained model[2] to calculate the ground truth of heatmap and have applied mean square error loss between predicted heatmaps and ground truth. Fig. 12 shows the architectural diagram the of the keypoint heatmap predictor which takes the previous 5 frames and melspectrogram of audio signal and feed it to the model to predict the heatmaps of the frames. Hourglass architecture [57] is used after two layers of convolution which helps in generating better facial heatmaps.

In the experiments, we have used the 15 channel heatmaps and input and output sizes are (15, 96, 96). We have done the joint training of keypoint predictor architecture along with the generator architecture and fed the output of keypoint predictor architecture in the multimodal adaptive normalization network to learn the affine parameters and have optimized it with mean square error loss with the output of a pretrained model. The input of the keypoint predictor model is the previous 5 frames along with 256 audio mel spectrogram features which are concatenated along the channel axis. This is optimization with mean square error loss with a pretrained facial keypoint detection model.

[2] https://github.com/raymon-tian/hourglass-facekeypoints-detection

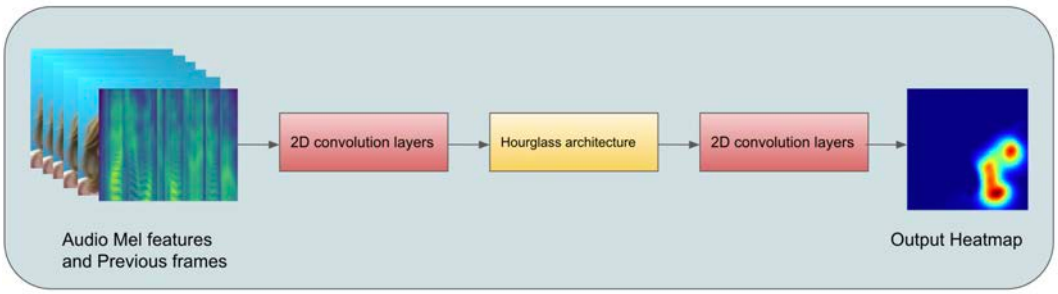


Fig. 12 – Keypoint heatmap predicted architecture

4.3.2 *Optical flow predictor*

The architecture is based on an encoder-decoder model (Fig. 13)to predict the optical flow of the next frame. We are giving the previous frames and current audio mel-spectrogram as an input to the model with KL loss and reconstruction loss. The pretrained model is then used in the generator to calculate the affine parameters. The input of the optical flow is previous 5 frames along with 256 audio melspectrogram features and is jointly trained along with the generator architecture and is optimized with mean square loss with the actual optical loss.

4.3.3 *Pitch extractor*

We extracted the pitch contour, F_0 using PyWorldVocoder tool [58] and quantized each frame to 256 possible values and encode them into a sequence of one-hot vectors as a pitch vector.

4.3.4 *Energy extractor*

We compute L2-norm of the amplitude of each Short-Time Fourier Transform (STFT) frame as the energy given by (Equation *(8)*) and then we add it to the expanded hidden sequence coming similar to pitch.

$$X(m,k) = \sum_{n=0}^{N-1} x[n-mH]w[n]exp(-2\pi ikn/N) \qquad (7)$$

where X(m,k) is the STFT of raw waudio waveform x[n] with window w[n] and m is the frame index , $k \in [0:K]$ and for every frame,m there are $K+1$ spectral vectors.

$$Energy(m) = \left(\sum_{k=0}^{K}(|x(m,k)|)^2\right)^{1/2} \qquad (8)$$

4.3.5 *Audio melspectrogram extractor*

We transfer the raw waveform into melspectrograms by setting the frame size and hop size to 1024 and 256 with respect to the sample rate of 22050 Hz.

4.4 Multiscale frame discriminator

We have used multiscale frame discriminator [59] to distinguish the fake and real image at the finer and coarser level. The class activation map-based layer is also used to distinguish the real or fake image by visualizing local and global attention maps. We have applied the adversarial loss (Equation *(14)*) on the information from the CAM output, n_{D_t} at different scale of the discriminator so that it will help the generator and discriminator to focus on local and global features and help in generating a more realistic image. This multiscale frame discriminator is based on Pix2PixHD [60].

$$L_{\text{cam}} = E_{y\sim P_t}[\log(n_{D_t}(y))] + E_{x\sim P_s}[\log(D(1-n_{D_t}(G(x))))] \qquad (9)$$

5. LOSSES

The proposed method is trained with different losses to generate realistic videos as explained below.

5.1 Adversarial loss

Adversarial loss is used to train the model to handle adversarial attacks and ensure the generation of high quality images for the video. The loss is defined as:

$$L_{\text{GAN}}(G,D) = E_{x\sim P_d}[\log(D(x))] + E_{z\sim P_z}[\log(D(1-G(z)))] \qquad (10)$$

where G tries to minimize this objective against an adversarial D that tries to maximize.

5.2 Reconstruction loss

Reconstruction loss [61] is used on the lower half of the image to improve the reconstruction in the mouth area. L1 loss is used for this purpose as described below:

$$L_{\text{RL}} = \sum_{n\epsilon[0,W]*[H/2,H]} (R_n - G_n) \qquad (11)$$

where, R_n and G_n are the real and generated frames respectively.

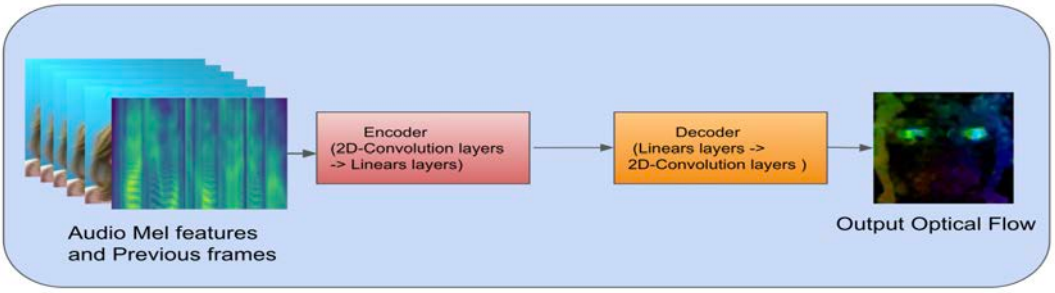


Fig. 13 – Optical flow predictor architecture

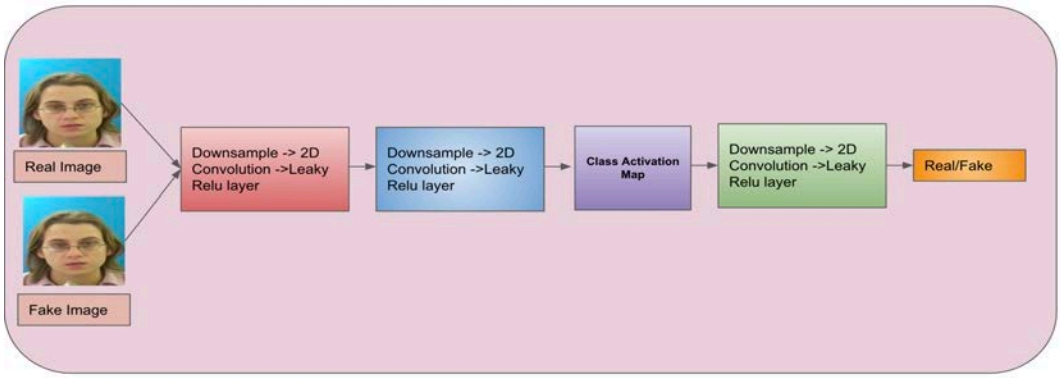


Fig. 14 – Discriminator architecture

5.3 Feature loss

Feature-matching loss [59] ensures the generation of natural-looking high quality frames. We take the L1 loss between generated images and real images for different scale discriminators and then sum it all. We extract features from multiple layers of the discriminator and learn to match these intermediate representations from the real and the synthesized image. This helps in stabilizing the training of the generator. The feature matching loss, $L_{FM}(G,D_k)$ is given by:

$$L_{\mathrm{FM}}(G, D_{\mathrm{k}}) = E_{(\mathrm{x,z})} \sum_{n=1}^{T} [\frac{1}{N_{\mathrm{i}}} ||{D_{\mathrm{k}}}^{(i)}(x) - {D_{\mathrm{k}}}^{(i)}(G(z))||_1] \quad (12)$$

where, T is the total number of layers and N_i denotes the number of elements in each layer.

5.4 Perceptual loss

The perceptual similarity metric is calculated between the generated frame and the real frame. This is done by using features of a VGG19 [62] model trained for ILSVRC classification and VGGFace [63] data set. The perceptual loss [64], (L_{PL}) is defined as:

$$L_{\mathrm{PL}} = \lambda \sum_{n=1}^{N} [\frac{1}{M_{\mathrm{i}}} ||F^{(i)}(x) - F^{(i)}(G(z))||_1] \quad (13)$$

where, λ is the weight for perceptual loss and $F^{(i)}$ is the ith layer of VGG19 network with M_i elements of VGG layer.

5.5 Class activation loss

We have used the class activation-based adversarial loss in the generator and discriminator which helps the model to learn local and global facial features and helps in cheek movement, blinks as well as image reconstruction.

$$L_{\mathrm{cam}} = E_{\mathrm{y}\sim \mathrm{P_t}}[\log(n_{\mathrm{D_t}}(y))] + E_{\mathrm{x}\sim \mathrm{P_s}}[\log(D(1 - n_{\mathrm{D_t}}(G(x))))] \quad (14)$$

where n_{D_t} is the class activation-based logits from the real and fake image.

5.6 Mean square loss

We have optimized the keypoint heatmap predictor and optical flow predictor using mean square loss between the generated keypoint heatmap and pretrained model [65] and generated optical flow and ground truth farneback [66] optical flow output.

6. QUALITY OF EXPERIENCE (QOE)

In order to avoid the spectators from quitting, thus increasing the revenue , the proposed model is able to control the quality of experience. We derive our QoE model from [67]. Using subjective Mean Opinion Score (MOS) measurements, they derive QoE as a second degree function of the image PSNR and Frame Rate (FR), fitted to the MOS:

$$QoE = -8.97 + 0.056 \cdot \text{FR} + 0.41 \cdot \text{PSNR} - 0.0038 \cdot \text{PSNR}^2 - 0.001 \cdot \text{FR}^2 + 0.00079 \cdot \text{FR} \cdot \text{PSNR} \quad (15)$$

Knowing the average PSNR and frame size, we use this model to calculate each receiver's QoE at present and estimate their QoE in the future for different profiles.
The total QoE for each receiver, which aims to reflect their satisfaction with the whole video streaming experience, will be a function of the individual QoE corresponding to each player.
The QoE metric has several advantages:

- Due to erratic network connectivity or low bandwidth, the Quality of Experience (QoE) can be low. With the proposed model we can significantly improve the QOE by sending the audio signal and synthesizing the video at the receiver's end, thus improving the PSNR.
- The proposed video streaming pipeline helps in dynamically using the proposed video generation architecture when the quality of experience goes below the threshold PSNR level. It thus gives the flexibility to control the QOE based on the compute resource, bandwidth availability and importance of the speaker in the video conference.

7. EXPERIMENTS

7.1 Implementation details

7.1.1 Data sets

We have used the GRID [68], LOMBARD GRID [69], Crema-D [70] and VoxCeleb2 [71] data sets for the experiments and evaluation of different metrics.

GRID: GRID [68] is a large multi-talker audiovisual sentence corpus to support joint computational-behavioral studies in speech perception. In brief, the corpus consists of high quality audio and video (facial) recordings of 1000 sentences spoken by each of the 34 talkers (18 male, 16 female). Sentences are of the form "put red at G9 now".

LOMBARD GRID: Lombard GRID [69] is a bi-view audiovisual Lombard speech corpus that can be used to support joint computational-behavioral studies in speech perception. The corpus includes 54 talkers, with 100 utterances per talker (50 Lombard and 50 plain utterances). This data set follows the same sentence format as the audiovisual GRID corpus, and can thus be considered as an extension of that corpus.

CREMA-D: CREMA-D [70] is a data set of 7,442 original clips from 91 actors. These clips were from 48 male and 43 female actors between the ages of 20 and 74 coming from a variety of races and ethnicities (African American, Asian, Caucasian, Hispanic, and Unspecified). Actors spoke from a selection of 12 sentences. The sentences were presented using one of six different emotions (Anger, Disgust, Fear, Happy, Neutral, and Sad) and four different emotion levels (Low, Medium, High, and Unspecified).

VOXCELEB2: VoxCeleb2 [71] is a very large-scale audio-visual speaker recognition data set collected from open-source media. Voxceleb2 contains over 1 million utterances for over 6,000 celebrities, extracted from videos uploaded to YouTube. The data set is fairly gender balanced, with 61 % of the speakers male.

7.1.2 Preprocessing steps

Videos are processed at 25fps and frames are resized into 256X256 size and audio features are processed at 16khz. The ground truth of optical flow is calculated using the farneback optical flow algorithm [66]. To extract the keypoint heatmaps, we have used the pretrained hourglass face keypoint detection [65]. Every audio frame is centered around a single video frame. To do that, zero padding is done before and after the audio signal and use the following formula for the stride.

$$stride = \frac{\text{audio sampling rate}}{\text{video frames per sec}}$$

We extract the pitch, F0 using using PyWorldVocoder [72] from the raw waveform with the frame size of 1024 and hop size of 256 sampled at 16khz to obtain the pitch of each frame and compute the L2-norm of the amplitude of each STFT frame as the energy. We quantize the F0 and energy of each frame to 256 possible values and encode them into a sequence of one-hot vectors as p and e respectively and then feed the value of p, e and 256 dimensional melspectrogram features in the proposed normalization method.

7.1.3 Metrics

To quantify the quality of the final generated video, we use the following metrics. Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM), Cumulative Probability Blur Detection (CPBD) and Average Content Distance (ACD). PSNR, SSIM, and CPBD measure the quality of the generated image in terms of the presence of noise, perceptual degradation, and blurriness respectively. ACD [44] is used for the identification of the speaker from the generated frames by using OpenPose [73]. Along with image quality metrics, we also calculate Word Error Rate (WER) using pretrained LipNet architecture [74], Blinks/sec using [75] and Landmark Distance (LMD) [76] to evaluate our performance of speech recognition, eye-blink reconstruction and lip reconstruction respectively.

1. PSNR- Peak Signal to Noise Ratio: It computes the peak signal to noise ratio between two images. The higher the PSNR the better the quality of the reconstructed image.

2. SSIM- Structural Similarity Index: It is a perceptual metric that quantifies image quality degradation. The larger the value the better the quality of the reconstructed image.

3. CPBD- Cumulative Probability Blur Detection: It is a perceptual based no-reference objective image sharpness metric based on the cumulative probability of blur detection developed at the image.

4. WER- Word error rate: It is a metric to evaluate the performance of speech recognition in a given video. We have used LipNet architecture [74] which is pretrained on the GRID data set for evaluating the WER. On the GRID data set, Lipnet achieves 95.2 percent accuracy which surpasses the experienced human lipreaders.

5. ACD- Average Content Distance ([44]): It is used for the identification of speakers from the generated frames using OpenPose [73]. We have calculated the Cosine distance and Euclidean distance of representation of the generated image and the actual image from Openpose. The distance threshold for the OpenPose model should be 0.02 for Cosine distance and 0.20 for Euclidean distance [77]. The lesser the distances the more similar the generated and actual images.

6. LMD - Landmark Distance ([76]): To ensure realistic and accurate lip movement, ensuring good performance on speech recognition we use this metric. We calculate the landmark points [78] on both real and generated images at the scale of 256*256 and use the lip region points i.e., points 49-68 and call then as LR and LF respectively. LR refers to lip region from ground truth image and LF corresponds to lip region from generated/fake image. T is the number of frames. Then, we calculate the euclidean distance between each corresponding pairs of landmarks on LR and LF. The LMD is defined as:

$$LMD = \frac{1}{T} * \frac{1}{P} \sum_{t=1}^{T} \sum_{p=1}^{P} ||LR_{t,p} - LF_{t,p}|| \qquad (16)$$

7. Blinks/sec: To capture the blinks in the video, we are calculating the blinks/sec so that we can better understand the quality of animated videos. Fig. 15 shows the 6 points which are used to calculate the Eye Aspect Ratio (EAR) given in Equation *(17)*. We have used SVM and eye landmarks along with Eye Aspect Ratio (EAR) used in Real-Time Eye Blink Detection using Facial Landmarks [75] to detect the blinks in a video.

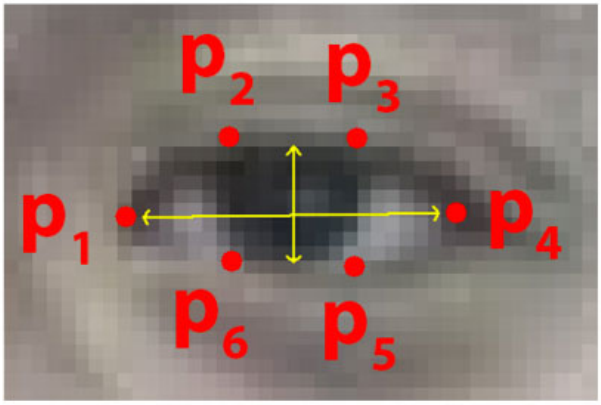


Fig. 15 – Description of the 6 eye points

$$EAR = \frac{||p2 - p6|| + ||p3 - p5||}{||p1 - p4||} \qquad (17)$$

7.1.4 Training and inference

Our model is implemented in pytorch and takes approximately 7 days to run on 4 Nvidia V100 GPUs for training. In the training stage, the model is trained with a multiscale frame discriminator with adversarial loss (Equation *(6)*), class activation map-based loss (Equation *(14)*) and feature matching loss (Equation *(12)*). The generator is trained with adversarial loss (Equation *(6)*), class activation map-based loss (Equation *(14)*), reconstruction loss (Equation *(11)*), perceptual loss (Equation 5.4), and key-point predictor/optical flow-based mean square error loss are also used to ensure generation of natural-looking, high quality frames.
We have taken the Adam optimizer [79] with learning rate = 0.002 and β_1= 0.0 and β_2 = 0.90 for the generator and discriminators.

7.2 Implementation results

7.2.1 Quantitative results

Tables 1,2,3,4 compare the proposed method with its competitors and shows better SSIM, PSNR, CPBD, Word Error Rate (WER), blinks/sec and LMD on GRID [68], Crema-D [70], GRID-Lombard [69] and Voxceleb2 [71] data sets, suggesting highly expressive and realistic video synthesis. The proposed method has shown superior results on most of the metrics in all the mentioned data sets.

7.2.2 QoE metric

We have computed the QoE metric for various data sets using Equation *(15)*. For our experiments we have taken the 25fps for synthesizing the video. Table 5 shows the QoE metric for various data sets when synthesizing the video from audio using the proposed method. The higher the QoE metric is, the better the model is. We can dynamically control the QoE based on the need of the video conferencing and during erratic network conditions.

Table 1 – Comparison of the proposed method(MAN-keypoint and MAN-optical) with other previous works for GRID data set

Method	SSIM↑	PSNR↑	CPBD↑	WER↓	ACD-C↓	ACD-E↓	blinks/sec	LMD↓
FOMM[80]	0.833	26.72	0.214	38.21	0.004	0.088	0.56	**0.718**
OneShotA2V[4]	0.881	28.571	0.262	27.5	0.005	0.09	0.15	0.91
RSDGAN[42]	0.818	27.100	0.268	**23.1**	-	$1.47x10^{-4}$	0.39	-
Speech2Vid[39]	0.720	22.662	0.255	58.2	0.007	$1.48x10^{-4}$	-	-
ATVGnet[48]	0.83	**32.15**	-	-	-	-	-	1.29
X2face[43]	0.80	29.39	-	-	-	-	-	1.48
CascadedGAN[46]	0.81	27.1	0.26	23.1	-	$1.47x10^{-4}$	0.45	-
MAN-optical	**0.908**	29.78	**0.272**	23.7	**0.005**	**$1.41x10^{-4}$**	0.45	**0.77**
MAN-keypoint	0.887	29.01	0.269	25.2	0.006	$1.41x10^{-4}$	0.48	0.80

Table 2 – Comparison of the proposed method(MAN-keypoint and MAN-optical) with other previous works for CREMA-D data set

Method	SSIM↑	PSNR↑	CPBD↑	WER↓	ACD-C↓	ACD-E↓	blinks/sec	LMD↓
FOMM[80]	0.654	20.74	0.186	NA	0.007	0.12	-	1.041
OneShotA2V[4]	0.773	24.057	0.184	NA	0.006	0.96	-	0.632
RSDGAN[42]	0.700	23.565	0.216	NA	-	$1.40x10^{-4}$	-	-
Speech2Vid[39]	0.700	22.190	0.217	NA	0.008	$1.73x10^{-4}$	-	-
MAN-optical	0.826	27.723	0.224	NA	0.004	$1.62x10^{-4}$	-	0.592
MAN-keypoint	**0.841**	**28.01**	**0.228**	NA	**0.003**	**$1.38x10^{-4}$**	-	**0.51**

Table 3 – Comparison of the proposed method(MAN-keypoint and MAN-optical) with other previous works for GRID Lombard data set

Method	SSIM↑	PSNR↑	CPBD↑	WER↓	ACD-C↓	ACD-E↓	blinks/sec	LMD↓
FOMM[80]	0.804	22.97	0.381	NA	0.003	0.078	0.37	1.09
OneShotA2V[4]	0.922	28.978	0.453	NA	0.002	0.064	0.1	0.61
Speech2Vid[39]	0.782	26.784	0.406	NA	0.004	0.069	-	0.581
MAN-optical	0.895	26.94	0.43	NA	0.001	0.048	0.21	0.588
MAN-keypoint	**0.931**	**29.62**	**0.492**	NA	**0.001**	**0.046**	**0.31**	textbf0.563

Table 4 – Comparison of the proposed method(MAN-keypoint and MAN-optical) with other previous works for VOXCELEB2 data set

Method	SSIM↑	PSNR↑	CPBD↑	WER↓	ACD-C↓	ACD-E↓	blinks/sec	LMD↓
OneShotA2V[4]	0.698	20.921	0.103	NA	0.011	0.096	0.05	0.72
MAN-optical	0.714	21.94	0.118	NA	0.008	0.067	0.21	0.65
MAN-keypoint	**0.732**	**22.41**	**0.126**	NA	**0.004**	**0.058**	**0.28**	**0.47**

Table 5 – Average QoE on proposed method

Method	QoE ↑
MAN-optical(GRID)	**1.232**
MAN-keypoint(GRID)	1.074
MAN-optical(GRID-Lombard)	0.624
MAN-keypoint(GRID-Lombard)	**1.20**
MAN-optical(CREMA-D)	0.797
MAN-keypoint(CREMA-D)	**0.860**
MAN-optical(VoxCeleb2)	-0.595
MAN-keypoint(VoxCeleb2)	**-0.472**

7.2.3 Qualitative results

Expressive aspect: Fig. 16 displays the lip synchronized frames of a speaker speaking the word 'bin' and 'please' as well as the blinking of the eyes. Fig. 17 shows the comparison of the proposed model with previous work [52] where the proposed model shows better image reconstruction and lip synchronization. The generated videos are given at [3].

Fig. 16 – Top: The speaker speaking the word 'bin' , Middle : The speaker speaking the word 'please', Bottom: The speaker blinking his eyes

[3] https://sites.google.com/view/itu2021

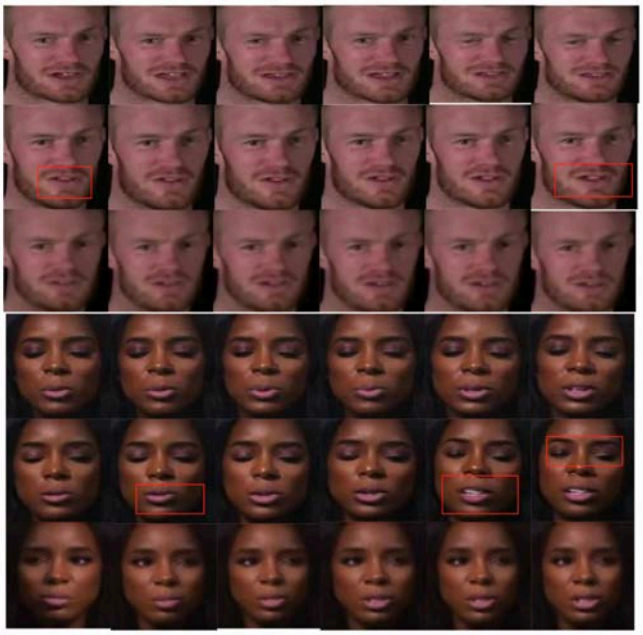


Fig. 17 – Top: Actual frames of voxceleb2 [71] data set , Middle : Predicted frames from proposed method, Bottom: Predicted frame from [52]

Architecture analysis: Fig. 18 shows the optical flow map and class activation-based heatmaps at different expressions of the speakers while speaking. The optical flow map has a different color while speaking and the opening of eyes as compared to closing of mouth and the blinking of eyes. The CAM-based heatmap shows the attention regions in the heatmap which captures the local as well as global features during video generation. The bottom part of the figure shows the predicted keypoints from the keypoint predictor calculated using the max operator to find the coordinates of the maximum value in each predicted heatmap (15, 96, 96).

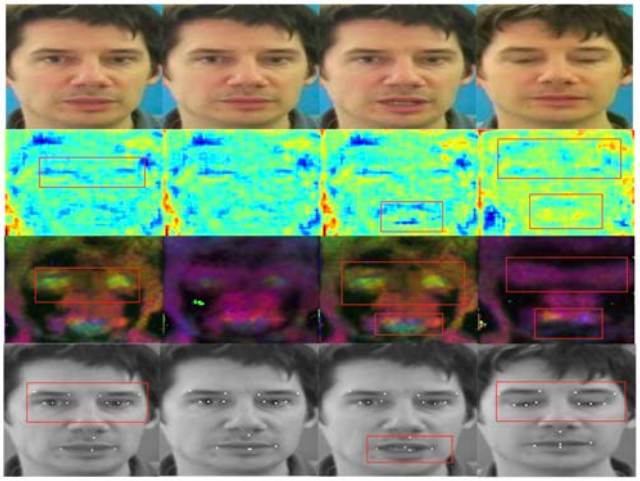


Fig. 18 – Top: The speaker with different expressions, Middle1 : CAM-based attention map, Middle2: Predicted optical flow from the optical flow generator architecture, Bottom: Predicted Key-points from Keypoint predictor architecture

Eye blinks: The average human blink rate of 0.28 blinks/second, especially when considering that the blink rate increases to 0.4blinks/second during conversation. Fig. 19 shows the sharp decline in the eye aspect ratio [75] at the centre which justifies the generation of blinks in the predicted videos. Table 1 shows the blinks/sec of 0.45 on the GRID data set.

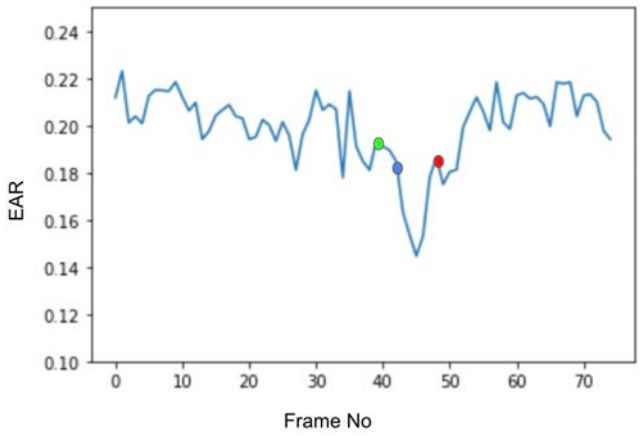


Fig. 19 – A blink is detected at the location where a sharp drop occurs in the EAR signal (blue dot). We consider the start (green dot) and end (red dot) of the blink to correspond to the peaks on either side of the blink location (Color figure online).

Comparison with video to video synthesis architecture: We have compared the proposed method with First Order Motion Model (FOMM) for image animation [80] on the GRID data set which generates the video sequences so that an object in a source image is animated according to the motion of a driving video. The comparison is done to see how effectively driving audio signals instead of driving video helps in reconstructing the expressive video as shown in Fig. 20. Tables 1, 2, 3 compare the various metrics between FOMM and the proposed model and show better image reconstruction metrics (SSIM, PSNR, CPBD,LMD) and WER but FOMM has more blinks/sec as compared to the proposed method. The reason for better WER is a limited number of utterances in the GRID data set and faster speaking style of the speaker which the proposed method is better able to capture as compared to FOMM.

7.3 Ablation study

To study the effectiveness of the proposed model and its novel multimodal adaptive normalization approach. We have shown that multimodal adaptive normalization is flexible to incorporate the various architecture shown in Section 7.3.1 and its effectiveness in the generation of realistic videos.We have also studied the incremental effect of audio and video features such as optical flow, melspectrogram, pitch and energy in Section 7.3.2.

Fig. 20 – Top: Actual frames of speaker of GRID data set. Middle: Predicted frames from proposed method with keypoints predicted from keypoint predictor. Bottom: Predicted frames from the FOMM method [80]

7.3.1 Network analysis in multimodal adaptive normalization

We have done the ablation study on three architectures, namely 2D convolution, partial 2D convolution [81, 82] and 2D convolution+Efficient Channel Attention (ECA) [83] for extracting video features and two architectures namely 1D convolution and LSTM for audio features as shown in Fig. 10 and Fig. 11 to study its effect on multimodal adaptive normalization with optical flow predictor in the proposed method. Table 6 shows that 2DConv+ECA+LSTM has improved the reconstruction metrics such as SSIM, PSNR and CPBD as well as word error rate and blinks/sec as compared to other networks. The image quality reduced with the use of partial 2D convolution which demonstrates that since the predicted optical flow is dense, holes in the optical flow has some spatial relation with other regions which are better captured by other networks.

Table 6 – Ablation study of different networks of multimodal adaptive normalization on GRID data set

Method	SSIM↑	PSNR↑	CPBD↑	blinks/sec	WER↓
2DConv+1dConv	0.875	28.65	0.261	0.35	25.6
Partial2DConv+1dConv	0.803	28.12	0.256	0.15	29.4
2DConv+ECA+1dConv	0.880	29.11	0.263	0.42	23.9
2DConv+LSTM	0.896	29.25	0.086	0.260	24.1
Partial2DConv+LSTM	0.823	28.12	0.258	0.12	28.3
2DConv+ECA+LSTM	0.908	29.78	0.272	0.45	23.7

7.3.2 Incremental effect of multimodal adaptive normalization

We study the incremental effect of multimodal adaptive normalization of the proposed model with the Optical Flow Predictor (OFP) and 2DConv+ECA+LSTM combination in multimodal attention normalization on a GRID data set. Table 7 shows the impact of the addition of melspectrogram features, pitch, predicted optical flow in multimodal adaptive normalization. The base model consists of generator and discriminator architecture with a static image in the adaptive normalization.

Table 7 – Incremental study of multimodal adaptive normalization on GRID data set

Method	SSIM↑	PSNR↑	CPBD↑	blinks/sec	WER↓
Base Model(BM)	0.776	27.99	0.213	0.02	57.9
BM + OFP+mel	0.878	28.43	0.244	0.38	27.4
BM + OFP+mel+pitch	0.881	28.57	0.264	0.41	24.1
BM+OFP+mel+pitch+energy	0.908	29.78	0.272	0.45	23.7

8. PSYCHOPHYSICAL ASSESSMENT

Results are visually rated (on a scale of 5) individually by 25 persons, on three aspects, lip synchronization, eye blinks and eyebrow raises and quality of video on a GRID data set. The subjects were shown anonymous videos at the same time for the different audio clips for side-by-side comparison. Table 8 clearly shows that MAN-based proposed architecture performs significantly better in quality and lip synchronization which is of prime importance in videos.

Table 8 – Psychophysical evaluation (in percentages) based on users rating on GRID daatset

Method	Lip-Sync↑	Eye-blink↑	Quality↑
MAN	91.8	**90.5**	**79.6**
OneShotA2V[4]	90.8	88.5	76.2
RSDGAN[42]	**92.8**	90.2	74.3
Speech2Vid[39]	90.7	87.7	72.2

9. TURING TEST

To test the naturalism of the generated videos we conduct an online Turing test on a GRID data set [4]. Each test consists of 20 questions with 10 fake and 10 real videos. The user is asked to label a video real or fake based on the aesthetics and naturalism of the video. Approximately 300 user data is collected and their score of the ability to spot fake video is displayed in Fig. 21.

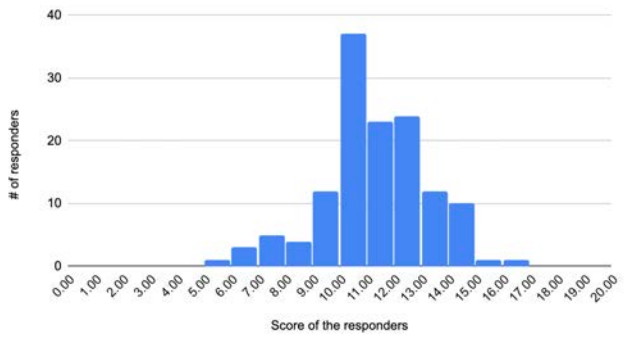


Fig. 21 – Distribution of user scores for the online Turing test

[4] https://forms.gle/DM1DRcTToQFvUpTa7

10. CONCLUSIONS AND FUTURE WORK

We have seen that the proposed video streaming pipeline with multimodal adaptive normalization-based architecture to generate the video helps in reducing the network bandwidth in unreliable Internet conditions. The proposed video streaming pipeline can control the quality of experience based on the compute resource and bandwidth availability. It helps in data privacy by synthesizing the video on the avatar of that person.

Although this implementation provides a proof of concept for the underlying idea, further work is needed to implement a full body low latency, low bandwidth video streaming environment to further enhance the quality of experience. With the rapid improvement of hardware capabilities in mobiles and personal computers, this is unlikely to be a major obstacle. As evidenced by the recent announcement of the NVIDIA Maxine project [3], hurdles are surmountable and these ideas can be translated into a practical system that provides immense gains over the conventional methods.

REFERENCES

[1] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. "Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol". In: *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 267–282. ISBN: 978-1-939133-01-4. URL: https://www.usenix.org/conference/nsdi18/presentation/fouladi.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[3] Sid Sharma. "AI Can See Clearly Now: GANs Take the Jitters Out of Video Calls". In: *NVIDIA Blog* (Aug. 2020).

[4] Neeraj Kumar, Srishti Goel, Ankur Narang, and Mujtaba Hasan. "Robust One Shot Audio to Video Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[5] Johannes L. Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 4104–4113.

[6] S. Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, S. Wang, and Shanshe Wang. "Image and Video Compression With Neural Networks: A Review". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30 (2020), pp. 1683–1698.

[7] Chaochao Lu and X. Tang. "Surpassing Human-Level Face Verification Performance on LFW with GaussianFace". In: *AAAI*. 2015.

[8] Z. Huang, Xiaowei Zhao, S. Shan, R. Wang, and X. Chen. "Coupling Alignments with Recognition for Still-to-Video Face Recognition". In: *2013 IEEE International Conference on Computer Vision* (2013), pp. 3296–3303.

[9] N. Kumar, P. Belhumeur, and S. Nayar. "FaceTracer: A Search Engine for Large Collections of Images with Faces". In: *ECCV*. 2008.

[10] I. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, S. Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *NIPS*. 2014.

[11] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: (Feb. 2015).

[12] Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15 (2014), pp. 1929–1958.

[13] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: (July 2016).

[14] Xun Huang and Serge Belongie. "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization". In: Oct. 2017, pp. 1510–1519. DOI: 10.1109/ICCV.2017.167.

[15] Hyeonseob Nam and Hyo-Eun Kim. *Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks*. May 2018.

[16] Junho Kim, Minjae Kim, Hyeon-Woo Kang, and Kwanghee Lee. *U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation*. July 2019.

[17] Jimmy Ba, Jamie Kiros, and Geoffrey Hinton. "Layer Normalization". In: (July 2016).

[18] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. "Semantic Image Synthesis with Spatially-Adaptive Normalization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[19] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. *World-Consistent Video-to-Video Synthesis*. July 2020.

[20] Peter de Rivaz and Jack Haughton. "Av1 bitstream & decoding process specification". In: *The Alliance for Open Media* (2018), p. 182.

[21] *Versatile Video Coding (VVC)*. https://jvet.hhi.fraunhofer.de/. Accessed: 2020-10-27.

[22] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. "Blazeface: Sub-millisecond neural face detection on mobile gpus". In: *arXiv preprint arXiv:1907.05047* (2019).

[23] Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, and Matthias Grundmann. "Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs". In: *arXiv preprint arXiv:1907.06724* (2019).

[24] Adrian Bulat and Georgios Tzimiropoulos. "How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks)". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1021–1030.

[25] Ivan Grishchenko, Artsiom Ablavatski, Yury Kartynnik, Karthik Raveendran, and Matthias Grundmann. "Attention Mesh: High-fidelity Face Mesh Prediction in Real-time". In: *arXiv preprint arXiv:2006.10962* (2020).

[26] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. "BlazePose: On-device Real-time Body Pose tracking". In: *arXiv preprint arXiv:2006.10204* (2020).

[27] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. "Deep high-resolution representation learning for human pose estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 5693–5703.

[28] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 269–286.

[29] Peter Eisert and Bernd Girod. "Analyzing facial expressions for virtual conferencing". In: *IEEE Computer Graphics and Applications* 18.5 (1998), pp. 70–78.

[30] Peter Eisert. "MPEG-4 facial animation in video analysis and synthesis". In: *International Journal of Imaging Systems and Technology* 13.5 (2003), pp. 245–256.

[31] A. Simons and Stephen Cox. "Generation of mouthshapes for a synthetic talking head". In: *Proceedings of the Institute of Acoustics, Autumn Meeting* (Jan. 1990).

[32] FirstName Alpher, FirstName Fotheringham-Smythe, and FirstName Gamow. "Can a machine frobnicate?" In: *Journal of Foo* 14.1 (2004), pp. 234–778.

[33] Andreea Stef, Kaveen Perera, Hubert Shum, and Edmond Ho. "Synthesizing Expressive Facial and Speech Animation by Text-to-IPA Translation with Emotion Control". In: Dec. 2018, pp. 1–8. DOI: 10.1109/SKIMA.2018.8631536.

[34] Tero Karras, Timo Aila, Samuli Laine, Antti Herva, and Jaakko Lehtinen. "Audio-driven facial animation by joint end-to-end learning of pose and emotion". In: *ACM Transactions on Graphics* 36 (July 2017), pp. 1–12. DOI: 10.1145/3072959.3073658.

[35] Sarah Taylor, Moshe Mahler, Barry-John Theobald, and Iain Matthews. "Dynamic units of visual speech". In: July 2012, pp. 275–284.

[36] Pif Edwards, Chris Landreth, Eugene Fiume, and Karan Singh. "JALI: an animator-centric viseme model for expressive lip synchronization". In: *ACM Transactions on Graphics* 35 (July 2016), pp. 1–11. DOI: 10.1145/2897824.2925984.

[37] Wesley Mattheyses and Werner Verhelst. "Audiovisual speech synthesis: An overview of the state-of-the-art". In: *Speech Communication* 66 (Nov. 2014). DOI: 10.1016/j.specom.2014.11.001.

[38] Supasorn Suwajanakorn, Steven Seitz, and Ira Kemelmacher. "Synthesizing Obama: learning lip sync from audio". In: *ACM Transactions on Graphics* 36 (July 2017), pp. 1–13. DOI: 10.1145/3072959.3073640.

[39] Joon Son Chung, Amir Jamaludin, and Andrew Zisserman. "You said that?" In: *British Machine Vision Conference*. 2017.

[40] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. "Temporal Generative Adversarial Nets with Singular Value Clipping". In: Oct. 2017. DOI: 10.1109/ICCV.2017.308.

[41] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Generating Videos with Scene Dynamics". In: (Sept. 2016).

[42] Konstantinos Vougioukas, Stavros Petridi, and Maja Pantic. "End-to-End Speech-Driven Facial Animation with Temporal GANs". In: *Journal of Foo* 14.1 (2004), pp. 234–778.

[43] O. Wiles, A.S. Koepke, and A. Zisserman. "X2Face: A network for controlling face generation by using images, audio, and pose codes". In: *European Conference on Computer Vision*. 2018.

[44] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. "MoCoGAN: Decomposing motion and content for video generation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1526–1535.

[45] R. Yi, Zipeng Ye, J. Zhang, H. Bao, and Yongjin Liu. "Audio-driven Talking Face Video Generation with Learning-based Personalized Head Pose". In: *arXiv: Computer Vision and Pattern Recognition* (2020).

[46] Dipanjan Das, Sandika Biswas, Sanjana Sinha, and Brojeshwar Bhowmick. "Speech-Driven Facial Animation Using Cascaded GANs for Learning of Motion and Texture". In: (Oct. 2019).

[47] Chelsea Finn, P. Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *ICML*. 2017.

[48] Lele Chen, Ross Maddox, Zhiyao Duan, and Chenliang Xu. *Hierarchical Cross-Modal Talking Face Generationwith Dynamic Pixel-Wise Loss*. May 2019.

[49] Lele Chen, Zhiheng Li, Ross Maddox, Zhiyao Duan, and Chenliang Xu. "Lip Movements Generation at a Glance". In: July 2018.

[50] Hang Zhou, Y. Liu, Z. Liu, Ping Luo, and X. Wang. "Talking Face Generation by Adversarially Disentangled Audio-Visual Representation". In: *AAAI*. 2019.

[51] K Prajwal, Rudrabha Mukhopadhyay, Vinay Namboodiri, and C Jawahar. *A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild*. Aug. 2020.

[52] Hao Zhu, Huaibo Huang, Y. Li, A. Zheng, and R. He. "Arbitrary Talking Face Generation via Attentional Audio-Visual Coherence Learning." In: *arXiv: Computer Vision and Pattern Recognition* (2020).

[53] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[54] Alireza M. Javid, Sandipan Das, M. Skoglund, and S. Chatterjee. "A ReLU Dense Layer to Improve the Performance of Neural Networks". In: *ICASSP*. 2021.

[55] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. "Learning Deep Features for Discriminative Localization." In: *CVPR* (2016).

[56] Dmitry Nikitko. *stylegan-encoder. https://github.com/Puzer/stylegan-encoder*. 2019.

[57] Gary Storey, Ahmed Bouridane, Richard Jiang, and Chang-Tsun Li. "Atypical Facial Landmark Localisation with Stacked Hourglass Networks: A Study on 3D Facial Modelling for Medical Diagnosis". In: Jan. 2020, pp. 37–49. ISBN: 978-3-030-32582-4. DOI: 10.1007/978-3-030-32583-1_3.

[58] PyWORLD. "https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder". In: (2019).

[59] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[60] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[61] Yanchun Li, Nanfeng Xiao, and Wanli Ouyang. "Improved Generative Adversarial Networks with Reconstruction Loss". In: *Neurocomputing* 323 (Oct. 2018). DOI: 10.1016/j.neucom.2018.10.014.

[62] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv 1409.1556* (Sept. 2014).

[63] Wang Mei and Weihong Deng. "Deep Face Recognition: A Survey". In: (Apr. 2018).

[64] Alexandre Alahi Justin Johnson and Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: (2016).

[65] facial keypoint detection. "https://github.com/raymon-tian/hourglass-facekeypoints-detection". In: (2017).

[66] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: vol. 2749. June 2003, pp. 363–370. DOI: 10.1007/3-540-45103-X_50.

[67] Saman Zadtootaghaj, Steven Schmidt, and Sebastian Möller. "Modeling gaming QoE: Towards the impact of frame rate and bit rate on cloud gaming". In: *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2018, pp. 1–6.

[68] FirstName Alpher and FirstName Fotheringham-Smythe. "Frobnication revisited". In: *Journal of Foo* 13.1 (2003), pp. 234–778.

[69] Najwa Alghamdi, Steve Maddock, Ricard Marxer, Jon Barker, and Guy J. Brown. *A corpus of audio-visual Lombard speech with frontal and profile view, The Journal of the Acoustical Society of America 143, EL523 (2018); https://doi.org/10.1121/1.5042758*. 2018.

[70] Houwei Cao, David Cooper, Michael Keutmann, Ruben Gur, Ani Nenkova, and Ragini Verma. "CREMA-D: Crowd-sourced emotional multimodal actors dataset". In: *IEEE transactions on affective computing* 5 (Oct. 2014), pp. 377–390. DOI: 10.1109/TAFFC.2014.2336244.

[71] J. S. Chung, A. Nagrani, and A. Zisserman. "VoxCeleb2: Deep Speaker Recognition". In: *INTERSPEECH*. 2018.

[72] PyWorldVocoder. "https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder". In: (2017).

[73] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields". In: *arXiv preprint arXiv:1812.08008*. 2018.

[74] Yannis M Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. "LipNet: End-to-End Sentence-level Lipreading". In: *GPU Technology Conference* (2017). URL: https://github.com/Fengdalu/LipNet-PyTorch.

[75] T. Soukupová and Jan Cech. "Real-Time Eye Blink Detection using Facial Landmarks". In: 2016.

[76] Lele Chen, Zhiheng Li, Ross K. Maddox, Zhiyao Duan, and Chenliang Xu. "Lip Movements Generation at a Glance". In: (2018). arXiv: 1803.10404 [cs.CV].

[77] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris Metaxas. *Learning to Forecast and Refine Residual Motion for Image-to-Video Generation*. 2018.

[78] D.E. King. "Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research". In: (2009).

[79] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[80] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. "First Order Motion Model for Image Animation". In: (2020). arXiv: 2003.00196 [cs.CV].

[81] Guilin Liu, Kevin J. Shih, Ting-Chun Wang, Fitsum A. Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Catanzaro. "Partial Convolution based Padding". In: *arXiv preprint arXiv:1811.11718*. 2018.

[82] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. "Image Inpainting for Irregular Holes Using Partial Convolutions". In: *The European Conference on Computer Vision (ECCV)*. 2018.

[83] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. *ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks*. Oct. 2019.

AUTHORS

Neeraj Kumar is currently working as Senior Machine Learning Scientist at Hike Private Limited, India. He is also a PHD student at Indian Institute of Technology, Delhi, India and completed his B-tech(ECE) from Indian Institute of Technology, Kharagpur, India. His current area of interest is multimodal AI along with the foundation and theoretical aspect of machine learning. He has published papers in top conferences such as CVPR, Interspeech & IJCAI.

Dr. Ankur Narang is currently the VP of AI technologies at Hike Private Limited. He holds a B.Tech. & Ph.D. from IIT Delhi in CSE and has 40+ publications in top international computer science machine learning conferences and journals, along with 15 granted US patents. He was one amongst Top 10 data scientists in India in 2017 (Analytics India Mag) in recognition of solid scientific and industry contributions to the field of data science and artificial intelligence. In 2018, he was given the Top 50 Analytics Award at the MachineCon conference in recognition of exemplary leadership and contributions to ML/AI (Analytics India Magazine). He was also conferred Top 100 Innovative CIO Award in 2019, for distinguished leadership in innovative technologies based digital transformation (CIO Axis). In 2002, he was awarded Sun Microsystem's prestigious "Innovation Leadership Award" for significant contributions to the Hardware Acceleration Project.

Dr. Brejesh Lall is a professor at Electrical Engineering Department at IIT Delhi and has contributed to research & teaching in the general area of signal processing. He is the head of Bharti School of Telcom Technology and Management, and the coordinator of two centers of excellence, viz. Airtel IIT Delhi Centre of Excellence in Telecommunications and Ericsson IIT Delhi 5G Center of Excellence. He is also the incharge of an IoT laboratory that he set up in collaboration with Samsung. Besides this, he is the NCC co-ordinator of IIT Delhi. The areas in which he has been publishing and doing sponsored research are centered on signal processing. The areas include object representation, tracking and classification, odometry, depth map generation, representation and rendering. He is also exploring vector sensor-based underwater acoustic communications, and performance issues in molecular communications. He has mentored 5 startups, in the areas of virtualization, geofencing, UAV based solutions and recommendations and data mining. He actively participates in building and deploying technology. He has also served as an expert in numerous government and private agencies in aspects related to signal processing.

Nitish Kumar Singh is currently working in the area of speech, text and vision. He has completed his post graduation from Manipal Institute of Technology, Bangalore.

A DYNAMIC Q-LEARNING BEAMFORMING METHOD FOR INTER-CELL INTERFERENCE MITIGATION IN 5G MASSIVE MIMO NETWORKS

Aidong Yang, Ph.D[1], Xinlang Yue[1], Mohan Wu, Ph.D[1], Ye Ouyang, Ph.D[1]
[1]Telecom Artificial Intelligence Lab, AsiaInfo Technologies, Beijing, China

NOTE: Corresponding author: Aidong Yang, Ph.D, yangad@asiainfo.com

Abstract – *Beamforming is an essential technology in 5G Massive Multiple-Input Multiple-Output (MMIMO) communications, which are subject to many impairments due to the nature of wireless transmission channel. The Inter-Cell Interference (ICI) is one of the main obstacles faced by 5G communications due to frequency-reuse technologies. However, finding the optimal beamforming parameter to minimize the ICI requires infeasible prior network or channel information. In this paper, we propose a dynamic Q-learning beamforming method for ICI mitigation in the 5G downlink that does not require prior network or channel knowledge. Compared with a traditional beamforming method and other industrial Reinforcement Learning (RL) methods, the proposed method has lower computational complexity and better convergence efficiency. Performance analysis shows the quality of service improvement in terms of Signal-to-Interference-plus-Noise-Ratio (SINR) and the robustness towards different environments.*

Keywords – 5G beamforming, inter-cell interference, massive MIMO, reinforcement learning

1. INTRODUCTION

Massive Multiple-Input Multiple-Output (MMIMO) technology in 5G is a competent solution that significantly improves system capacity, signal coverage and spectral-efficiency by configuring hundreds of Antenna Elements (AEs) at the Base Station (BS) to shape effective beamforming [1, 2]. However, the quality of MMIMO beamforming depends on accurate Channel State Information (CSI), pilot contamination and ICI estimation [3]. Moreover, the MMIMO beamforming complexity becomes a challenge as the number of AEs at the BS increases. Therefore, it is necessary to explore an effective and efficient beamforming method for ICI mitigation with low power and low complexity [4].

In recent years, the accurate MMIMO beamforming has attracted extensive research [3, 4, 5, 6, 7], which almost follow two main directions: with and without CSI. Hybrid beamforming [3, 4, 5] is the representative of the former. It aims to reduce the expense of Radio Frequency (RF) chains and decrease the complexity of beamforming compared to conventional methods [2], but it needs to update beams frequently when pilots are received continually at the BS. A smart pilot assignment scheme, which is effective in mitigating interference but is aimed at a single cell, is proposed in [5] to reduce pilot contamination by smartly assigning orthogonal pilots to users. The latter mainly includes the first Monte Carlo (MC) method, which searches the optimal beamforming parameters but suffers from increasing computational complexity, and second supervised Deep Learning (DL) methods. One of them is reported in [7] to research the characters of wireless spatial channels and explore preferable pilot assignments for better channel estimation and beamforming, but supervised methods require model training beforehand and time-consuming sample data collection.

In this paper, an RL-assisted full dynamic beamforming method is developed to efficiently acquire the optimal beamforming parameters in the MMIMO system to address ICI issues. We fully consider the microcell and macro-cell multi-path transmission channels which present radio features with high user density and traffic loads focusing on pedestrian and vehicular users (Dense Urban-eMBB) scenarios [1, 2, 5], such as buildings, mountains and rivers, where the distribution of User Equipments (UEs) changes infrequently; these factors significantly impact coverage. To get optimal beamforming, firstly, we utilize a Poisson Point distribution model to estimate the occurrences of UE in the target cells with a long-term data statistical analysis; secondly, we apply an algorithm to fast search through huge volumes of parameters and obtain optimal values. Lastly, we send the optimal parameters into the BS beamforming simulator for the best SINR.

In summary, the main contribution of this work includes:

- The proposed RL beamforming method for an MMIMO system is meant to get the optimal beamforming parameter, such a method with multi-cell ICI is rarely discussed in literature. Besides, it does not need any prior network or channel information and it works for different UE distribution.
- Compared with the traditional beamforming method and other industrial RL methods, the proposed dynamic Q-learning method shrinks the action space during its process, thus it requires less time and computational complexity to operate.
- As proven in many simulation results, the proposed method performs better than the other methods. Moreover, it is robust to various starting states and different environments.

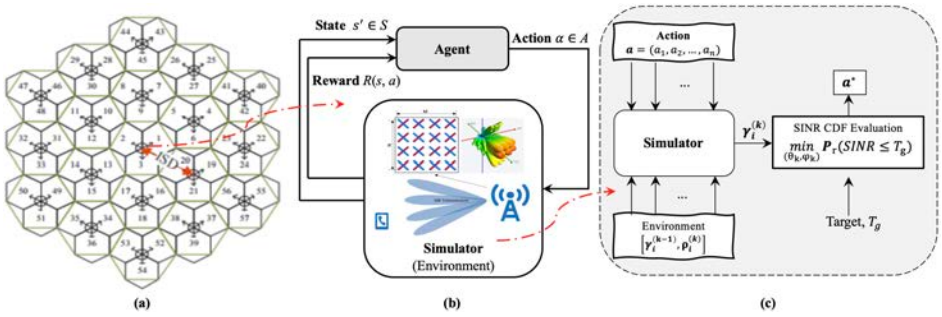


Fig. 1 – Illustration of the proposed RL-based beamforming (b) for MMIMO systems and the network layout (a) of 5G dense Urban-eMBB cells, in which the BS of target small cell #0 with $N_0(k)$ mobile users chooses the optimal beamforming $a_0^{*(k)}$ (c) to mitigate interference from the neighboring N_{cell} small cells at time slot k, and $N_0(k)$ users return estimated SINRs $\gamma_N(k)$ to the BS.

The rest of the paper is organized as follows. In Section 2, related work on RL-based ICI mitigation are presented. The system model and our proposed dynamic Q-learning scheme are presented in Section 3 and Section 4. In Section 5, simulation results are presented and the conclusion follows in Section 6.

2. RELATED WORK

ICI control is a key issue in 5G MMIMO systems, intensive research has been carried out to address this. Surveys have been carried out on ICI mitigation techniques in LTE downlink networks [8, 9], and research on ICI coordination techniques in 5G UFMC systems [10, 11].

RL-based approaches have been extensively applied in an ICI mitigation problem. For instance, a Q-learning-based power control scheme formulates the ICI coordination issue as a cooperative multi-agent control problem to improve the performance of the cellular systems is proposed in [12]. An RL-based power control scheme for ultra-dense small cells to improve network throughput and save energy consumption is presented in [13], in which the BS selects the downlink transmit power to manage interference. A dynamic RL-based ICI coordination algorithm as developed in [14] smartly offloads traffic to open access picocells and then improves the system throughput.

3. SYSTEM MODEL

ICI is caused by multiple sources transmitting signals with the same subcarrier and being received by a receiver. A user receives signals from the serving cell and neighboring cells but at different power levels due to the path loss.

3.1 AOA-based beamforming

The Angle-Of-Arrival(AOA)-based beamforming is usually used in 5G MMIMO systems, where the BS is configured with an antennas array composed of *W* AEs, and numbers of AEs are arranged as *M* per row and *L* per column [2].

In RF, the BS shapes beamforming for the k^{th} UE by configuring weights on AEs according to AOA $\langle \theta_k, \varphi_k \rangle$ [15], where θ_k is the azimuth and φ_k is the vertical angle of the k^{th} UE. The weights on the i^{th} AE in a row can be represented as

$$\omega_{ik} = e^{-j2\pi i d_h \sin\theta_k}, w_{ik} \in \mathcal{C}^{1*M} \quad (1)$$

where d_h is the row AE distance. And the l^{th} AE in the column can be obtained by

$$\xi_{lk} = e^{-j2\pi l d_v \cos\varphi_k}, \xi_{lk} \in \mathcal{C}^{L*1} \quad (2)$$

where d_v is the column AE distance. From *(1)* and *(2)*, the final beamforming weights for the k^{th} UE can be derived by

$$\Pi_k = \Psi_k \Omega_k \quad (3)$$

where

$$\begin{cases} \Omega_k = [\omega_{1k}, \omega_{2k}, \dots, \omega_{Mk}], \\ \Psi_k = [\xi_{1k}, \xi_{2k}, \dots, \xi_{Lk}]^T. \end{cases}$$

Since the final weights in *(3)* depend on $\langle \theta_k, \varphi_k \rangle$, the implementation complexity for $\langle \theta_k, \varphi_k \rangle$ estimation gets high as the perfect CSI needed, which is usually affected by ICI.

3.2 Search-based beamforming

To mitigate the ICI with a low complexity, a search-based beamforming algorithm is reported in [15], which uses MC to search the optimal weights rather than AOA estimation in *(3)*. In MC beamforming, the best weights are obtained by searching $\langle \theta_k, \varphi_k \rangle$ in all possible angles to minimize the ICI, i.e.

$$\langle \theta_k^*, \varphi_k^* \rangle \longleftarrow \arg \min_{\langle \theta_k, \varphi_k \rangle} P_r(SINR < T_g | h_j^{(k)}, \rho_j^{(k)}) \quad s.t. -\pi < \theta_k, \varphi_k \le \pi \quad (4)$$

where P_r is the probability of SINRs weaker than the target T_g given the channel $h_j^{(k)}$ and UE density $\rho_j^{(k)}$, and the SINR in (4) for the i^{th} UE located on the j^{th} cell can be expressed by [15]

$$SINR_{i,j} = \frac{p_{i,j} \varsigma_{i,j}^{-\nu}}{N_0 B + \sum_{k=1,k\neq j}^{N} p_k \varsigma_k^{-\nu}} \quad (5)$$

where ν is the path-loss exponent, $p_{.j}$ is the transmit power of the serving enode B_j, N is the number of neighboring enode B_s, p_k is the transmit power from B_s, $\varsigma_{.j}$ is the distance of the UE to the serving station, ς_k is the distance of the UE to each of the neighboring stations, and $N_0 B$ is the background noise with N_0 the thermal noise and B the system bandwidth.

According to [16], the UE density $\rho_0^{(k)}$ in *(4)* is assumed to follow the independently and identically distributed two-dimensional Poisson point process. The number of users $N_0^{(k)}$ of the target cell with area φ_0 is given by

$$Pr\{N_0^{(k)} = \lambda|\varphi_0\} = \frac{(\rho_0^{(k)}\varphi_0)^\lambda}{\lambda!} e^{-\rho_0^{(k)}\varphi_0} \qquad (6)$$

From *(4)* to *(6)*, the optimal parameters $\langle \theta^*, \varphi_k^* \rangle$ can be found, and the best weight Π_k can be derived by substituting *(4)-(6)* into *(1)-(3)*.

4. THE PROPOSED REINFORCEMENT LEARNING ASSISTED BEAMFORMING

Due to the lack of prior knowledge that is required to find the theoretical optimal solution of *(4)*, some research has been conducted over related surrogate RL optimization problems. Generally, apart from the MC method, Sarsa [17] and Q-learning [13] are attempted. Those methods lack convergence efficiency in practice even though their convergence can be guaranteed [18].

In this section, a dynamic Q-learning beamforming method is proposed to mitigate ICI and enhance convergence efficiency. Each BS exploits the user SINRs in a dense Urban-eMBB transmission environment and estimates the Probability Density Function (PDF) of users' occurences to achieve an optimal beamforming solution via trial without knowledge of the network and transmission channel.

4.1 RL-based beamforming

In the RL-based beamforming process as shown in Fig. 1(a)(b), the BS in the target cell estimates the probability density $\rho_0^{(t)}$ of users' occurrences in the target small cell $\#0$ by a long-term data statistical analysis in *(6)* at time slot t. Once all served users send SINRs $\gamma^{(t-1)}$ at the time slot $(t-1)$ to the BS, the state $[\rho_0^{(t)}, \gamma^{(t-1)}]$ observed by the BS at the time slot t is obtained, and then an RL-based beamforming algorithm is applied for searching the optimal parameters for the ICI mitigation and coverage optimization. We formulate the beamforming optimization problem under the MMIMO system context as an RL problem and therefore provide a dynamic Q-learning scheme to address the issue.

First, we define the agent to be the MMIMO system, the set of states $S \triangleq \{s_l\}_{l=0}^{m-1}$ to be the levels of average regional SINR, the set of actions $A \triangleq \{a_j\}_{j=0}^{n-1}$ to be the possible combinations of antenna parameters. More precisely, each s_l is an interval of the SINR value, s_0 is the optimal SINR value interval, i.e. the highest achievable SINR value derived from expert experiences in the current environment. Similarly, s_{m-1} is the lowest SINR range. And as l increases, the boundary values of s_l decreases, thus a higher l implies poorer signal performance state s_l. Each action a_j is an antenna parameter's choice made by the MMIMO system, and consists of azimuth, vertical angle and beam width. The environment is a signal simulator, see Section 5.1 for more detail. The objective is to approach the optimal target SINR state s_0 to achieve the best signal performance. It covers the probability in *(4)* of average regional SINR given by the simulator and guided by selected action a. The environment (Fig. 1(c)) grants the agent a reward $r_{s,a}$ after the latter takes an $a \in A$ when it is in $s \in S$.

Formally, we denote the state-action value function, the expected discounted reward, as $Q(s, a)$. In the table $\mathbf{Q} \in R^{m \times n}$, we use notation [19] $Q(s, a) \triangleq [\mathbf{Q}]_{s,a}$ and update entries by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_{s,a} + \delta \max_{a'} Q(s', a') - Q(s,a)] \qquad (7)$$

where $\alpha : 0 < \alpha < 1$ is the learning rate and $\delta : 0 < \delta < 1$ is the *discount* factor and determines the importance of future rewards. s' and a' are the next state and action, respectively.

An *episode* is a period of time in which an interaction between the environment and the agent takes place. Here, an episode is of (at most) τ transitional discrete time step t. During an episode $i : i \in \{0, 1, \ldots, \zeta\}$, the agent makes a decision to maximize the effects of actions decided by itself. To achieve this goal, we apply the ϵ- greedy learning strategy to balance exploration and exploitation, where $1 - \epsilon : 0 < \epsilon < 1$ is the exploration rate and serves as the threshold probability to select a random $a \in A$, as opposed to selecting an action based on exploitation. To add randomness, the ϵ increases in every episode from ϵ_{min} until it reaches a preset upper bound.

The S space is constructed by partitioning the range of the process Cumulative Distribution Function (CDF), which is the probability of users with SINR under the given T_g in *(4)*. The components of A space is shown in Table 1. Through a finite series of $a \in A' := A - \mathbf{C}$ (will be discussed later), the agent attempts to approach s_0 in response to simulated s_i at step t within an episode.

Table 1 – Learning Parameters

Parameter	Value
Learning rate α	0.01
Reward decay rate δ	0.9
Minimum exploration rate ϵ_{min}	0.9
Number of episodes ζ	22
Number of steps in each episode τ	40
Number of states	30
Number of actions	855

4.2 Reward signals

4.2.1 Reward design with Q-initialization

As discussed in [20], reward signals in our simulation environment are crucial to the RL Markov Decision Process (MDP) since agents are expected to learn the optimal policy under industrial criteria.

Since adding additional rewards follows the policy invariance [20], the reward function $r(s,a)$ within our problem setting consists of two main parts:

$$r(s,a) = r(s_0,a)_{goal} + r(s,a)_{inter} \tag{8}$$

$r(s_0,a)_{goal}$ is given to the agent if s_0 is approached and $r(s,a)_{inter}$ works as intermediate reward in the training process when $s \neq s_0$.

We aim to construct reward shaping for $r(s,a)_{inter}$ using the potential-based method to help guide the agent in MDP; the potential-based shaping function is defined as [20]:

Definition 1 *Let any S, A, δ and any shaping reward function $F: S \times A \times S \to \mathbb{R}$ in MDP be given. F is **potential-based** if there exists a real-valued function $\Phi: S \to \mathbb{R}$ s.t.*

$$F(s,a,s') = \delta\Phi(s') - \Phi(s) \tag{9}$$

for all $s \neq s_0, s' \in S, a \in A$.

Therefore, based on the results in [20], such an F can guarantee consistency with the optimal policy that the agent learned. Luckily, there is no need to construct the shaping function from scratch [21], since the design of F is equivalent to the initialization of $[\mathbf{Q}]_{s,a}$.

Suppose the optimal policies learnt in our model with and without potential-based F are π' and π, respectively. Let initial Q function of π be $Q(s,a) = Q_0(s,a)$ with shaping rewards $\delta\Phi(s') - \Phi(s)$, and initial Q function of π' be $Q'(s,a) = Q_0(s,a) + \Phi(s)$ with no shaping rewards.

By *(7)*, we have the update error:

$$\begin{cases} Q_{error} = r_{s,a} + \delta\Phi(s') - \Phi(s) + \delta \max\limits_{a'} Q(s',a') - Q(s,a) \\ Q'_{error} = r_{s,a} + \delta \max\limits_{a'} Q'(s',a') - Q'(s,a) \end{cases} \tag{10}$$

and now insert ΔQ and $\Delta Q'$, the difference between current and initial values of Q and Q' respectively, into the update error:

$$\begin{cases} \Delta Q(s,a) = Q(s,a) - Q_0(s,a) \\ \Delta Q'(s,a) = Q'(s,a) - Q_0(s,a) - \Phi(s) \end{cases} \tag{11}$$

we have

$$\begin{aligned} Q_{error} &= r_{s,a} + \delta\Phi(s') - \Phi(s) + \delta \max_{a'}(Q_0(s',a') \\ &\quad + \Delta Q(s',a')) - Q_0(s,a) - \Delta Q(s,a) \\ &= r_{s,a} + \delta \max_{a'}(\Phi(s') + Q_0(s',a') + \Delta Q(s',a')) \\ &\quad - Q_0(s,a) - \Delta Q(s,a) - \Phi(s) \\ &= r(s,a) + \delta \max_{a'} Q'(s',a') - Q'(s',a') \\ &= Q'_{error} \end{aligned} \tag{12}$$

Therefore, we investigate the relationship between r_{inter} and $Q_0(s,a)$ to decide the form of r_{inter}. In the MDP problem setting [18], the discounted return from time step t is $G_t = \sum_{k=0}^{\infty} \delta^k r_{t+k+1}$, and since $\delta \in (0,1)$, if r_{inter} is formed as a bounded series based on the distance from s_0 to s_l: $r(s,a)_{inter} \leq r_{bound}$, where $r_{bound} \leq 1$, we have

$$\begin{aligned} G_t &= \sum_{k=0}^{\infty} \delta^k r_{t+k+1} \\ &\leq \sum_{k=0}^{\infty} \delta^k r_{bound} \\ &\leq r_{bound} \sum_{k=0}^{\infty} \delta^k \\ &= \frac{r_{bound}}{1-\delta} \end{aligned} \tag{13}$$

then for optimal policy π' [18]

$$\max_a Q^{\pi'}(s,a) = E[G_t] \leq r_{goal} \tag{14}$$

we know r_{inter} and r_{goal} satisfy:

$$\frac{r_{bound}}{1-\delta} \leq r_{goal} \tag{15}$$

(15) gives an explicit gap between the two parts of $r(s,a)$ and also directly influences the following initialization of $Q(s,a)$.

4.2.2 Q-initialization setting

We rewrite the initial Q table of policy π' and the final converged table as $Q_0^{\pi'}$ and $Q_{final}^{\pi'}$ respectively. By (*(7)*),

$$\begin{aligned} Q^{\pi'}(s,a) &\leftarrow Q_0^{\pi'} + \alpha(r_{bound} + \delta Q_0^{\pi'} - Q_0^{\pi'}) \\ &= Q_0^{\pi'} + \alpha(1-\delta)(Q_{final}^{\pi'} - Q_0^{\pi'}) \end{aligned} \tag{16}$$

we can derive that

$$Q_0^{\pi'} > Q_{final}^{\pi'} = \frac{r_{bound}}{1-\delta} \tag{17}$$

to guarantee the convergence of the model update. And under the Q-learning scheme, *(17)* always provide chances of exploration for actions that have not been attempted.

In this end, we give reward signals for $r(s,a)$ as follows:

$$r_{s_l,a} := \begin{cases} -\frac{e^{0.1\cdot(l-2)}}{e^{2.8}+1} & s_l \geq s_2 \\ -\frac{0.01}{e^{2.8}+1} & s_l = s_1 \\ \frac{e^{0.2\cdot(30-i)}}{e^{2.8}+1} & s_l = s_0 \end{cases} \quad (18)$$

here $s_l \geq s_2$ means $l \geq 2$ and set $r_{bound} = -\frac{0.01}{e^{2.8}+1}$ from *(18)* to follow the conditions we derived in *(13)*, *(15)*. Therefore, we can initialize the Q function as $[\mathbf{Q}]_{s,a} := \mathbf{0}_{|S|\times|A|}$ to satisfy *(17)*.

Algorithm 1 Optimal Action Selection Control

Input: Initial CDF state s_{init} and target state s_0.
Output: Optimal a to approach s_0 during episode i.

Define customized S, A, ϵ and δ.
Initialize $\mathbf{C} := \{\}$, $\boldsymbol{Q} := \mathbf{0}_{|S|\times|A|}$, $i := 0$
Initialize $s := s_{init}$, $t := 0$
repeat
  while $t < \tau$ **do**
    $\epsilon := \max_{a'}(\epsilon_{min}, \epsilon_{min} + \delta \cdot t/(\tau \cdot \zeta))$
    Sample $k_1, k_2 \sim \mathcal{U}(0,1)$
    if $k_1 \leq \epsilon$ **then**
      if $k_2 > \delta(1-\epsilon)$ **then**
        Select $a \in A - \mathbf{C}, a = \arg\max_{a'} Q(s,a')$
      else
        Select $a \in A, a = \arg\max_{a'} Q(s,a')$
      end
    else
      Select $a \in A - \mathbf{C}$ randomly
    end
    Perform a in the simulator obtain $s', r(s,a)$
    Update the entry $Q(s,a)$ as in *(7)*
    $s \leftarrow s', t \leftarrow t+1$
    if $s \neq s_0$ **then**
      Append a in $\mathbf{C}$
    else
      Early stopping
      return a
    end
  end while
until $s = s_0$ otherwise proceed to episode $i+1$

4.3 Dynamic Q-Learning algorithm

Considering the computational and equipment cost in an MMIMO system, the delaying effect of reward should be minimized. Then after each step t, we use twice ϵ-greedy strategy, the controller to help avoid the action that is unrelated to s_0 to dynamically shrink the A space in order to make up for the delay in *(18)*. Therefore, the controller plays a highly efficient role as the penalty signal in our reward and serves as a reinforced mechanism to assist the selection. The upper bound of the time complexity for the dynamic Q-learning method is in $\mathcal{O}(mn)$ [22].

For a total of at most n trials in ζ episodes with a fixed initial environment setting, algorithm 1 will stop training the agent once s_0 is approached rather than continuing the process due to the reward signals design in our model:

Controller **C**: As shown in Algorithm 1, controller **C** will shrink the action space related to s in every step t based on the double ϵ- greedy principle. This operation enables the optimal action selection with higher and higher probability as t goes on.

Reward $r_{s,a}$: *(18)* guarantees the agent learns a global optimum, our target action, instead of continuously jumping on some local optimum for meaningless rewarding [23].

Reward signals and controller **C** attempt to guide the agent by avoiding redundant scoring and long term penalties. The agent itself continuously updates the learning policy under the guidance of both of them.

4.4 Other existing methods

For the not too large $S \times A$ space defined in Section 4.1, the MC exhaustion algorithm often serves as a baseline solution for the problem in Section 3. It requires testing on all possible $a \in A$ to ensure the best action among space A.

Therefore, we apply classical model-free RL methods: Q-learning (off-policy) and Sarsa (on-policy) [18] in this problem setting. They differ mainly in the Q function updating style, while Q-learning holds (*(7)*), Sarsa follows the update below:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_{s,a} + \delta Q(s',a') - Q(s,a)] \quad (19)$$

Parameters for these models are set the same as in Table 1. Unsurprisingly, off-policy based methods are superior to on-policy methods [18] in the experiment discussed later.

With the experience gained from Algorithm 1, Algorithm 2 is proposed to test the trained agent's policy with any randomized given s_{init}.

5. SIMULATIONS AND DISCUSSIONS

To thoroughly investigate the performance of the proposed RL-assisted full dynamic beamforming method and validate the effectiveness of the theoretical analysis previously, we present statistical results of SINRs and computational complexity of the proposed algorithm compared to other industrial methods. We implement Algorithm 1 within the environment below with preset parameters shown in both Table 1 and Table 2.

Table 2 – Environment components

Simulation Parameter	Value	Simulation Parameter	Value
Antenna 3dB-Bandwidth in Azimuth (°)	$15 \sim 110$	Number of Observed UEs K_0	100
Antenna 3dB-Bandwidth in Elevation (°)	$0 \sim 30$	Receiver Bandwidth (MHz)	20
Antenna Tilt Angle (°)	$-3 \sim 15$	Receiver Height (m)	1.5
Carrier Frequency (GHz)	3.5	MM Array Type	URA
Height of BS (m)	25	MM Array Size	8×8
Transmit Power (dBm)	44	MM Mechanical Downtilt	15

Algorithm 2 Evaluation Algorithm

Input: Target state s_0 and $\boldsymbol{Q}$ from Algorithm 1.
Output: Optimal a, target s with rewards for Z episodes.

Load the experienced $\boldsymbol{Q} := [\mathbf{Q}]_{s,a}$, $z := 0$
repeat
  Randomize s_{init}
  Choose $a = \arg\max_{a'} Q(s_{init}, a')$
  Perform a in the simulator and obtain $s', r_{s_{init}}, t$
  Update $(s := s', a, r_{s_{init}}, t)_z$
  $z \leftarrow z + 1$
until $z = Z$

5.1 Environment setting

We use the three following metrics to set up the simulation environment and help compare:

- The simulation is based on the guidelines deined in [24] for evaluating 5G radio technologies in an urban macro-cell test environment which presents a radio channel with high user density and trafic loads focusing on pedestrian and vehicular users (Dense Urban-eMBB) [25].
- As shown in Fig. 1(a), the environment layout consists of 19 sites placed in a hexagonal layout, each with 3 cells, and the Inter-Site Distance (ISD) is 200 m.
- To visualize SINR for the simulation scenario we use the Close-In (CI) propagation path loss model [26], which calculates the path loss of transmitted power in 5G urban microcell and macro-cell scenarios. This model produces an RSRP (Reference Signal Receiving Power) map and a SINR map that shows reduced interference effects compared to other beamforming methods.

5.2 Computational complexity

The agent learns from the environment for 1000 epochs of all randomized s_{init} and stores policy experience in the Q-table described in Algorithm 1. In this stage, our model performs faster and is more stable than other methods mentioned above. We utilize the three following metrics to help compare:

Normalized Iteration Expectation $\mathcal{I}_E$: it indicates the scaled steps expectation to approach s_0 in 1000 epochs of training.

Computational Eficienc y (CE) $\mathcal{E}_*$: we deine the ratio below to relect computational cost saving:

$$\mathcal{E}_* \triangleq \frac{\mathcal{I}_{\mathrm{E}} \text{ for Baseline MC}}{\mathcal{I}_{\mathrm{E}} \text{ for method } i} \tag{20}$$

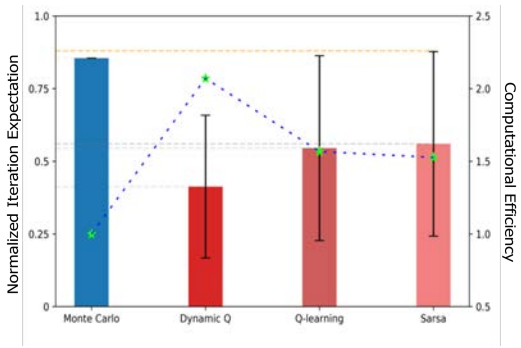
where $i \in \{\text{Dynamic Q, Q-learning, Sarsa}\}$.

Reward Scoring: This metric indicates how the dynamic Q method is different from other methods in speed and convergence when achieving reward.

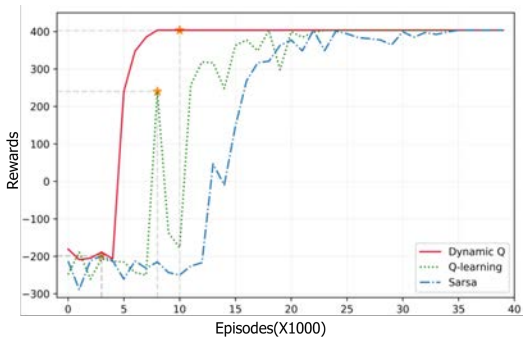
Fig. 2(a) displays the $\mathcal{I}_E$ with standard deviation, which implies stability in 1000 epochs, of how the dynamic Q model acts differently from Q-learning, Sarsa and MC. It takes the lowest normalized $\mathcal{I}_E$ needed to meet s_0 with the highest computational eficiency $\mathcal{E}_*$ (highlighted stars) and even doubles $\mathcal{E}_*$ compared to the baseline MC. (b) indicates the agility of our model in adapting to the environment. Given randomized s_{init}, the 95% conidence interval shadow indicates within 1000 epochs of training, the range and convergence rate of reward scoring for the dynamic Q model differs from other RL methods. Our model is able to fully train its agent in 10 episodes (without early stopping) with robustness and obtain the highest reward while other methods are still unstable under the two criteria.

5.3 SINR performance

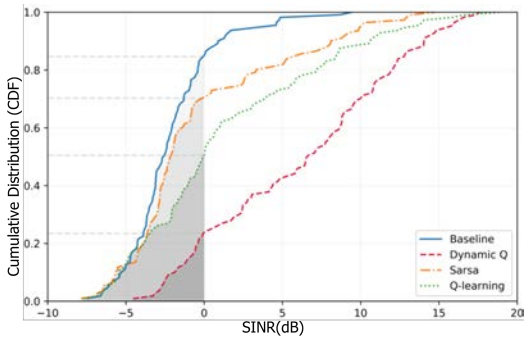
We show our model's shifting effect on SINR coverage in Fig. 2(c)(d) compared to other methods. With the optimal parameters derived from four models, respectively, within 10 episodes in (b) and sent into the simulator, (c) indicates ours is of the smallest weak SINR coverage that is lower than 0dB. The dynamic Q model suficientl y shifts the SINR coverage towards a strong SINR direction, and it enlarges the SINR coverage larger than 0 dB to over 50% of the total population in the Region of Interest (ROI). Specifically, (d) discloses that our model has the smallest probability density of users with weak SINR, for example, when SINR $\in (-5, 0]$, the probability is 23% with the dynamic Q model while it is 74%, 58%, 38% with the rest of the methods, respectively.



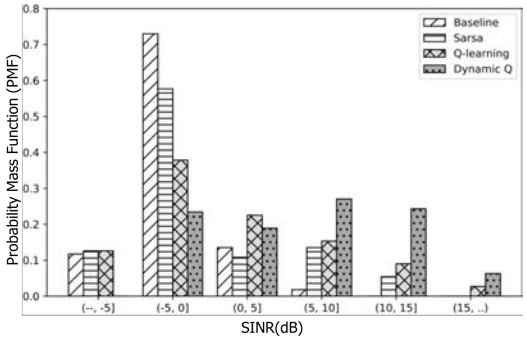
(a) The complexity of iteration expectation



(b) The best reward of ix ed iteration $N_{it} = 10$

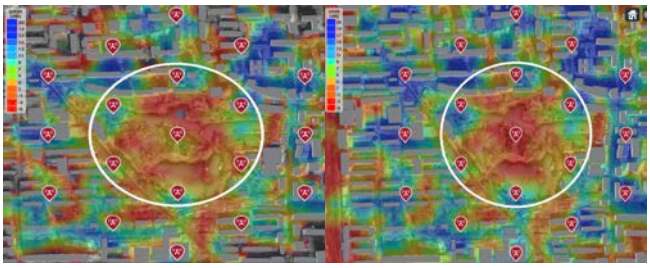


(c) The CDF of SINRs

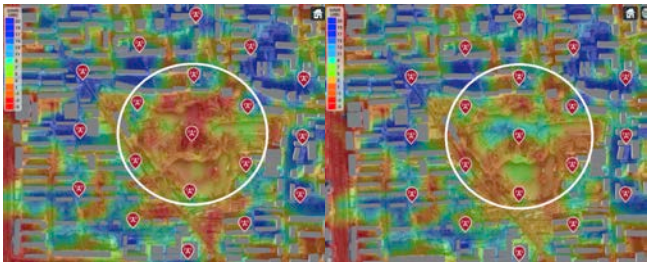


(d) The PMF of SINRs

Fig. 2 – Comparison of computational complexity and SINR improvements of the proposed dynamic Q algorithm with other industrial methods: MC (Baseline), Q-Learning and Sarsa. (a) shows the total iteration number for the optimal parameters; (b) displays their best reward when the iterate number is ix ed to $N_{\text{it}} = 10$, each point on the mean curve of rewards is averaged across 1000 epochs with random s_{init}, the shadow is the 95% conidence interval across 40 episodes of three models setting; (c) and (d) give the CDF and PMF of their SINRs.



(a) Baseline: $\bar{\gamma} = -2.036dB$ **(b)** SARSA: $\bar{\gamma} = -1.724dB$



(c) Q-learning: $\bar{\gamma} = 1.488dB$ **(d)** Dynamic Q $\bar{\gamma} = 6.319dB$

Fig. 3 – The average SINRs of different RL-based ICI mitigation algorithms in 5G MMIMO system, with parameters fed from Fig. 2(b). White circles are ROI. $W = 64$, $N_{cell} = 57$, $K_0 = 100$.

Fig. 3 displays the application when the optimal action is sent into the simulator of different models in 10 training episodes. The dynamic Q model is of the best average SINR of $\overline{\gamma} = 6.319$ dB in the ROI among all models.

In Table 3, we compare the average SINRs, across 6 different scenarios, for the dynamic Q model against MC, SARSA, and Q-Learning with parameters fed from Fig. 2(b). It is clear that the dynamic Q model improves the UE SINRs across 6 different environments, particularly in comparison with MC, where we achieve the average SINR improvements of around 8.3 dB, 10.4 dB, 12.2 dB 11.2 dB and 11.8 dB, respectively.

6. CONCLUSION

In this paper, we propose an RL (i.e. dynamic Q-learning) assisted full dynamic beamforming algorithm for the ICI mitigation in 5G MMIMO systems. This algorithm mitigates the ICI and reduces the computational complexity of the BS without knowledge of the network and transmission channel. Simulation results show the implementation complexity is lower and UE SINRs are signiicantly improved compared to other industrial methods. For example, in the dense Urban-eMBB scenario, the probability of weak SINRs in the target cell is about 60% lower and computational complexity is reduced by more than 50% compared to the benchmark.

Table 3 – Application scenarios

BS location		RL-based ICI mitigation algorithms			
Longitude	**Latitude**	**MC**	**Sarsa**	**Q-learning**	**Dynamic Q**
116.395659	39.959522	–2.036	–1.724	1.488	6.319
117.212147	39.161901	–4.732	–2.543	0.563	5.783
111.713038	40.832723	–7.931	–3.472	–1.239	4.374
111.710787	40.832027	–6.293	–2.174	0.897	4.978
111.709219	40.837586	–6.517	–2.573	1.296	5.381

REFERENCES

[1] Emil Björnson, Erik G Larsson, and Thomas L Marzetta. "Massive MIMO: Ten myths and one critical question". In: *IEEE Communications Magazine* 54.2 (2016), pp. 114–123.

[2] Jakob Hoydis, Stephan Ten Brink, and Mérouane Debbah. "Massive MIMO in the UL/DL of cellular networks: How many antennas do we need?" In: *IEEE Journal on selected Areas in Communications* 31.2 (2013), pp. 160–171.

[3] Xudong Zhu, Zhaocheng Wang, Linglong Dai, and Chen Qian. "Smart pilot assignment for massive MIMO". In: *IEEE Communications Letters* 19.9 (2015), pp. 1644–1647.

[4] Vishnu V Ratnam, Andreas F Molisch, Ozgun Y Bursalioglu, and Haralabos C Papadopoulos. "Hybrid beamforming with selection for multiuser massive MIMO systems". In: *IEEE Transactions on Signal Processing* 66.15 (2018), pp. 4105–4120.

[5] Xiaoguang Zhao, Elena Lukashova, Florian Kaltenberger, and Sebastian Wagner. "Practical hybrid beamforming schemes in massive mimo 5g NR systems". In: *WSA 2019; 23rd International ITG Workshop on Smart Antennas*. VDE. 2019, pp. 1–8.

[6] Deepak Mishra and Håkan Johansson. "Optimal channel estimation for hybrid energy beamforming under phase shifter impairments". In: *IEEE Transactions on Communications* 67.6 (2019), pp. 4309–4325.

[7] Kwihoon Kim, Joohyung Lee, and Junkyun Choi. "Deep learning based pilot allocation scheme (DL-PAS) for 5G massive MIMO system". In: *IEEE Communications Letters* 22.4 (2018), pp. 828–831.

[8] A Daeinabi, K Sandrasegaran, and X Zhu. "Survey of intercell interference mitigation techniques in LTE downlink networks". In: *Australasian Telecommunication Networks and Applications Conference (ATNAC) 2012*. IEEE. 2012, pp. 1–6.

[9] Beatriz Soret, Klaus I Pedersen, Niels TK Jørgensen, and Víctor Fernández-López. "Interference coordination for dense wireless networks". In: *IEEE Communications Magazine* 53.1 (2015), pp. 102–109.

[10] Shendi Wang, John S. Thompson, and Peter M. Grant. "Closed-Form Expressions for ICI/ISI in Filtered OFDM Systems for Asynchronous 5G Uplink". In: *IEEE Transactions on Communications* 65.11 (2017), pp. 4886–4898. DOI: 10.1109/TCOMM.2017.2698478.

[11] Reshma Ravindran and Abhishek Viswakumar. "Performance evaluation of 5G waveforms: UFMC and FBMC-OQAM with Cyclic Prefix-OFDM System". In: *2019 9th International Conference on Advances in Computing and Communication (ICACC)*. 2019, pp. 6–10. DOI: 10.1109/ICACC48162.2019.8986195.

[12] Mariana Dirani and Zwi Altman. "A cooperative reinforcement learning approach for inter-cell interference coordination in OFDMA cellular networks". In: *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*. IEEE. 2010, pp. 170–176.

[13] Hailu Zhang, Minghui Min, Liang Xiao, Sicong Liu, Peng Cheng, and Mugen Peng. "Reinforcement learning-based interference control for ultra-dense small cells". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 1–6.

[14] Meryem Simsek, Mehdi Bennis, and Ismail Güvenç. "Learning based frequency-and time-domain inter-cell interference coordination in HetNets". In: *IEEE Transactions on Vehicular Technology* 64.10 (2014), pp. 4589–4602.

[15] Joy Iong-Zong Chen, Bo Hueng Lee, and Wen Bin Wu. "Performance evaluation of BER for an Massive-MIMO with M-ary PSK scheme over Three-Dimension correlated channel". In: *Computers & Electrical Engineering* 65 (2018), pp. 196–206.

[16] Vladimir Poulkov, Pavlina Koleva, Oleg Asenov, and Georgi Iliev. "Combined power and inter-cell interference control for LTE based on role game approach". In: *Telecommunication Systems* 55.4 (2014), pp. 481–489.

[17] Aya Mostafa Ahmed, Alaa Alameer Ahmad, Stefano Fortunati, Aydin Sezgin, Maria Greco, and Fulvio Gini. "A Reinforcement Learning based approach for Multi-target Detection in Massive MIMO radar". In: *IEEE Transactions on Aerospace and Electronic Systems* (2021), pp. 1–1.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[19] Raghavendra V Kulkarni, Anna Fö rster, and Ganesh Kumar Venayagamoorthy. "Computational intel- ligence in wireless sensor networks: A survey". In: *IEEE communications surveys & tutorials* 13.1 (2010), pp. 68–96.

[20] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *ICML*. Vol. 99. 1999, pp. 278–287.

[21] Eric Wiewiora. "Potential-based shaping and Q-value initialization are equivalent". In: *Journal of Artiicial Intelligence Research* 19 (2003), pp. 205–208.

[22] Sven Koenig and Reid G Simmons. "Complexity analysis of real-time reinforcement learning". In: *AAAI*. 1993, pp. 99–107.

[23] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. "Inverse reward design". In: *Advances in neural information processing systems*. 2017, pp. 6765–6774.

[24] M Series. "Guidelines for evaluation of radio interface technologies for IMT-2020". In: (2017).

[25] M Series. "Guidelines for evaluation of radio interface technologies for IMT-Advanced". In: *Report ITU* 638 (2009), pp. 1–72.

[26] Shu Sun, Theodore S Rappaport, Timothy A Thomas, Amitava Ghosh, Huan C Nguyen, István Z Kovács, Ignacio Rodriguez, Ozge Koymen, and Andrzej Partyka. "Investigation of prediction accuracy, sensitivity, and parameter stability of large-scale propagation path loss models for 5G wireless communications". In: *IEEE Transactions on Vehicular Technology* 65.5 (2016), pp. 2843–2860.

AUTHORS

Aidong Yang received a Ph.D. degree in wireless communications from the Dalhousie University, Halifax, NS, Canada, in 2017. His research interests include wireless techniques, 5G communications, machine learning and its applications. He has authored and co-authored over 20 journal and conference papers.

Xinlang Yue received an M.S. degree in applied mathematics from Columbia University, New York, NY, US, in 2020. His research interests mainly focus on deep reinforcement learning algorithms with real-world applications.

Mohan Wu received a Ph.D. degree in numerical mathematics from University of Pittsburgh, Pittsburgh, PA, US, in 2019. His research interests mainly focus on the init e element method and machine learning algorithm.

Ye Ouyang, Ph.D., is CTO & Senior Vice President of AsiaInfo Technologies. Dr. Ouyang has distinguished experience in R&D and management in the telecommunications industry. Prior to AsiaInfo, Dr. Ouyang has been Verizon Fellow and Senior Manager in Verizon. His research is in the interdisciplinary area of wireless communications, data science, and AI. Dr. Ouyang has authored more than 30 academic papers, 40 patents, 10 international standards, and 8 books. Dr. Ouyang obtained a Ph.D. from Stevens Institute of Technology, a Master of Science from Tufts University, another Master of Science from Columbia University, and a Bachelor's degree of from Southeast University.

NETXPLAIN: REAL-TIME EXPLAINABILITY OF GRAPH NEURAL NETWORKS APPLIED TO NETWORKING

David Pujol-Perich[1], José Suárez-Varela[1], Shihan Xiao[2], Bo Wu[2], Albert Cabellos-Aparicio[1], Pere Barlet-Ros[1]
[1]Barcelona Neural Networking center, Universitat Politècnica de Catalunya., [2]Network Technology Lab., Huawei Technologies Co.,Ltd.

NOTE: Corresponding author: David Pujol-Perich, david.pujol.perich@upc.edu

Abstract – *Recent advancements in Deep Learning (DL) have revolutionized the way we can efficiently tackle complex optimization problems. However, existing DL-based solutions are often considered as black boxes with high inner complexity. As a result, there is still certain skepticism among the networking industry about their practical viability to operate data networks. In this context, explainability techniques have recently emerged to unveil why DL models make each decision. This paper focuses on the explainability of Graph Neural Networks (GNNs) applied to networking. GNNs are a novel DL family with unique properties to generalize over graphs. As a result, they have shown unprecedented performance to solve complex network optimization problems. This paper presents NetXplain, a novel real-time explainability solution that uses a GNN to interpret the output produced by another GNN. In the evaluation, we apply the proposed explainability method to RouteNet, a GNN model that predicts end-to-end QoS metrics in networks. We show that NetXplain operates more than 3 orders of magnitude faster than state-of-the-art explainability solutions when applied to networks up to 24 nodes, which makes it compatible with real-time applications; while demonstrating strong capabilities to generalize to network scenarios not seen during training.*

Keywords – AI/ML for networks, explainability, graph neural networks

1. INTRODUCTION

In recent years, Deep Learning (DL) has revolutionized the way we are able to solve a vast number of problems by finding meaningful patterns on large amounts of data. This acquired knowledge then enables us to make highly accurate predictions, leading to systematically outperforming state-of-the-art solutions in many different problems [1, 2]. However, in the field of networking, DL-based techniques still pose an important technological barrier to achieve market adoption. In general, Machine Learning (ML) solutions provide probabilistic performance guarantees, which typically degrade as the data deviates from the distribution observed during training. Moreover, neural networks have very complex internal architectures, often with thousands or even millions of parameters not interpretable by humans. As a result, they are treated as black boxes [3]. This limits the viability of these solutions to be applied to networks, as these are critical infrastructures where it is essential to deploy fully reliable solutions. Otherwise, a potential misconfiguration could lead to temporal service disruptions with serious economic damages for network operators.

In this vein, we do need mechanisms that can delimit the safe operational ranges of DL models. This makes it fundamental to *understand why and in what situations a DL-based solution can fail*. This can be achieved by producing human-readable interpretations of the decisions made by these models (e.g., interpret a routing decision given a traffic matrix and a network topology). This would not only enable us to achieve more mature and reliable DL solutions but also to enhance their performance by making ad-hoc adjustments for a particular network scenario (e.g., hyper-parameter tuning).

In this context, explainability solutions [4] have recently emerged as practical tools to interpret systematically the decisions produced by DL models. Particularly, these recently proposed solutions analyze trained DL models from a black-box perspective (i.e., they only analyze their inputs and outputs) and aim to discover which elements mainly drive the output produced by these models. As a result, they can eventually determine what are the most critical input elements to reach the final decisions. These kinds of techniques have been intensely examined in the field of computer vision, showing promising results [5].

At the same time, the last few years have seen the explosion of Graph Neural Networks (GNNs) [6], a new neural network family that has attracted large interest given its numerous applications to different fields where the information is fundamentally represented as graphs (e.g., chemistry [7], physics [8], biology [9], information science [10, 11]). This newly introduced mechanism has proven, to date, to be the only DL technique capable of generalizing with high accuracy to graphs of different sizes and structures not seen during the training phase.

In this context, GNNs have shown good properties to be applied in the field of computer networks, as many key components in network control and management problems are fundamentally represented as graphs (e.g., topology, routing). Indeed, we have already witnessed some successful GNN-based applications to network modeling and optimization [12, 13, 14, 15]. However, the fact that we are not able to understand the inner architecture of GNNs presents nowadays a major barrier that may hinder its adoption in real-world networks.

Explainability of GNNs has been recently explored in two main works. A first work emerging from the ML community [10] analyzes a well-known GNN model applied to a chemistry problem to quantify the impact of the different input elements (atoms, bonds) on the final model predictions (molecular properties). Likewise, the networking community has made a first attempt to apply a similar solution to several network optimization use cases [3]. However, both solutions are based on costly iterative optimization algorithms that need to be executed for each sample on which we want to obtain interpretations. Hence, they do not meet the requirements to make comprehensive analysis over large data sets and, more importantly, to be used in real-time applications. To address these limitations, this paper proposes NetXplain, a novel real-time *explainability solution for GNNs*. NetXplain introduces a novel approach where we use a GNN that learns, from Tabula Rasa, how to interpret the outputs of another GNN trained for a specific task (e.g., routing optimization). NetXplain produces human-understandable interpretations of GNNs comparable to those of state-of-the-art solutions [10, 3]. However, it makes this at a much more limited cost. In our evaluation, we apply NetXplain to RouteNet [12], a GNN model that predicts the per-path delay given a network snapshot as input (i.e., topology + routing + traffic matrix). To this end, we first train NetXplain on a data set with samples produced by Metis [3]. This training is done over a data set of limited size –5 to 10% of the original data set used in RouteNet [12]. Then, we validate the generalization power of our GNN-based method, by applying it to network scenarios fundamentally different from those seen during training. The evaluation results reveal the feasibility to train NetXplain over a small dataset produced by costly explainability solutions (e.g., [10, 3]), and be able to apply it over a wide variety of network scenarios. This eventually enables us to meet the needed requirements to make a comprehensive analysis of the safe operational range of GNN solutions at a limited cost. In this context, we show that NetXplain far outperforms state-of-the-art algorithms in terms of computational cost, running more than 3 orders of magnitude faster on average than Metis [3] when applied to samples of three real-world network topologies up to 24 nodes.

As an example, this explainability solution can be used as follows: given a GNN-based solution and a network scenario, NetXplain points to the particular network elements (e.g., devices, links, paths) that mostly affected the output decisions of the GNN model. This can be helpful for many different use cases, including: (i) test & troubleshooting of GNN-based solutions, (ii) reverse engineering, or (iii) improving network optimization solutions.

The remainder of this paper is structured as follows. First, Section 2 introduces the fundamental principles of GNNs and their application to networking. Then, Section 3 presents the related work on explainability for GNNs. In Section 5 we describe NetXplain, the proposed explainability solution. Afterward, Section 6 presents an evaluation of the accuracy and cost of NetXplain with respect to the state of the art. Finally, Section 7 presents a discussion on possible applications of the proposed explainability method, and Section 8 concludes the paper.

2. BACKGROUND

2.1 Graph neural networks

Graph neural networks are a novel neural network family designed to operate over graph-structured data, by capturing and modeling the inherent patterns in a graph. This has resulted in an unprecedented predictive power in many applications where data is structured as graphs. Despite the several variants of GNNs introduced in recent years, in this paper we focus on *Message-Passing Neural Networks (MPNN)* [7], as they represent a generic GNN framework.

MPNN operates over a graph *G*, in which nodes $v \in G$ are characterized with some initial features X_v. First, the hidden-state h_v^0 of each node $v \in G$ are initialized using their input node features X_v. Once each element v of the graph has its hidden-state h_v^0 initialized, they proceed to the message-passing phase, which shall be repeated a given number of times T. Fig. 1 illustrates how the message passing phase works. In each iteration t of the algorithm, every node v receives a message from each of its neighbors $u \in N(v)$. In MPNN, messages are generated using a function $m(\cdot)$ computed with the hidden state of the neighbor node. Then, once every node v has received the messages from its immediate neighbors, these messages are combined with an aggregation function $a(\cdot)$ producing a fixed-size output (e.g., an element-wise summation).

Finally, the algorithm reaches the update phase, in which nodes use the aggregated information received from their neighbors to update their own hidden states via the update function $u(\cdot)$.

Formally, the message passing at a given iteration t is defined as:

$$m_{v,j} = m(h_v^t,\ h_j^t,\ e_{v,j}) \quad (1)$$

$$M_v^{t+1} = a(\{m_{v,j} \mid j \in N(v)\}) \quad (2)$$

$$h_v^{t+1} = u(h_v^t,\ M_v^{t+1}) \quad (3)$$

In these equations, functions $m(\cdot)$ and $u(\cdot)$ can be computed through a universal function approximator, such as neural networks (e.g., feed-forward NN or recurrent NN). After T message passings, the hidden states of nodes typically converge to some fixed values [6]. Thus, these final hidden states pass through a readout function $r(\cdot)$ that computes the output of the GNN model. $r(\cdot)$ automatically learns the mapping from hidden-state representations to the output labels of the model y:

$$\hat{y} = r(h_v^T \mid v \in V) \quad (4)$$

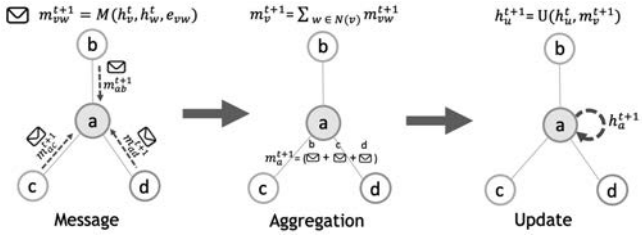


Fig. 1 – Message-passing phase: (left) Message, (mid) aggregation and (right) update.

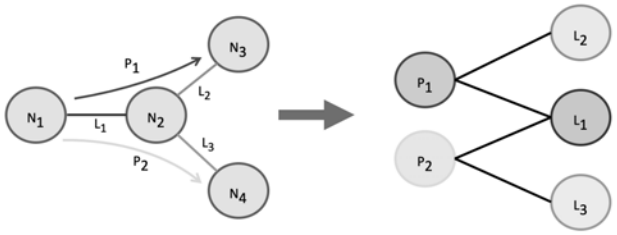


Fig. 2 – Transformation from the physical network scenario to the graph representation of RouteNet.

Such a function $r(\cdot)$ can also be implemented as a neural network, typically a feed-forward NN, and can be used to produce either node-level predictions by processing individually each node hidden state, or make global predictions of the graph by combining all the hidden states. In this latter case, hidden states are typically aggregated (e.g., element-wise sum) before they are introduced into the readout function.

This technology has proven to generalize successfully over graphs of different sizes and structures, which was not possible with traditional neural network architectures (e.g., feed-forward NN, convolutional NN, recurrent NN).

2.2 Graph neural networks applied to networking

The strong generalization capabilities of GNN over graphs make these models interesting for applications in the networking field since the most natural way to formalize many network control and management problems involves the use of graphs (e.g., topology, routing, interflow dependencies) [3]. Recently, several GNN-based solutions have been proposed to tackle different use cases in the field of computer networks (e.g., network modeling [12, 16], automatic routing protocols [13]). In this section, for illustrative purposes, we focus only on RouteNet [12], as it is quite representative of how GNN-based solutions represent and process network-related data to solve complex problems.

RouteNet targets the problem of modeling the per-path QoS metrics (e.g., delay, jitter) of a computer network. For this purpose, a network snapshot is provided as input: a network topology, a routing configuration, and a traffic matrix. To this end, this model makes a transformation of the physical network scenario into a more refined graph representation in which physical and logical elements are explicitly represented –*paths* and *links* in this case. More specifically, every *link* of the physical network topology is transformed into a node in the input graph of the GNN. Likewise, each source-destination path is also converted into a node. Finally, edges connect *links* with *paths* according to the routing configuration. Thus, each path is connected to those links that it traverses given the input routing scheme. This process is illustrated in Fig. 2, where we can observe how a physical network scenario with two paths and three links is transformed into the input graph of RouteNet. This graph representation enables us to model the complex relationships between the state of paths and links, and how they relate to the output per-path performance metrics (e.g., delay).

In this regard, applying explainability over this model would enable us to identify the most critical edges of its internal graph (i.e., path-link relations). We refer to critical edges as the set of path-link pairs that better explain the QoS metrics obtained by the model. Thus, with this solution, we can extract relevant knowledge of the processing made by the GNN given a network scenario, which can have many diverse applications, as later discussed in Section 7.

3. RELATED WORK

Recent years have attracted increasing interest in producing explainability solutions for neural network models (e.g., *Convolutional Neural Networks* [5]). Despite this, explainability techniques for GNN have been scarcely explored so far. In this context, GNNExplainer [17] is, to the best of our knowledge, the first proposal approaching this problem.

GNNExplainer is given as input a target GNN model and a sample graph $G = (V, E)$, with input features F. GNNExplainer, then, outputs a subset containing the connections $E' \subset E$ and the node features $F' \subset F$, that affect most critically the output of the target GNN (see Fig. 3). This is done by computing a set of weights W, formally defined in Eq. *(5)*, that represents how critical are the pair-wise connections of input graphs to the prediction accuracy of the target GNN.

$$W = \{w_{i,j} \mid (i,j) \in E\} \qquad (5)$$

Particularly, the most relevant connections are those that have more impact on the loss function used to train the model (e.g., mean squared error for regression tasks). The number of relevant connections produced by the algorithm can be tuned by setting a threshold on the resulting weights $w_{i,j} \in W$.

Overall, GNNExplainer is a generic solution proposed from the ML community that targets only at producing explainability representations of GNNs used for global graph classification, node-level classification, or link prediction. However, this solution does not support GNN-based models used for regression. In this context, a posterior solution proposed from the networking community presents Metis [3], a similar approach adapted to GNN models trained for regression problems, particularly showcasing its use in several networking applications.

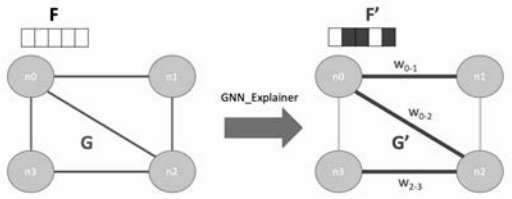


Fig. 3 – Schematic description of explainability solutions for GNN (e.g., GNNExplainer)

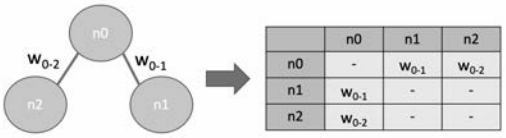


Fig. 4 – Explainability mask of an input graph.

Although GNNExplainer [17] and Metis [3] are able to produce quality explainability solutions for a vast range of problems, both have an important limiting factor. To compute E' and F' for each input sample, these solutions use an iterative convex optimization method, which is very time-consuming. For instance, producing a single explainability solution can take up to hundreds of seconds in scenarios with topologies between 14 and 24 nodes, as shown later in Section 6. This fact arguably prevents these methods to be applied for real-time operation. Moreover, their high cost makes them impractical to perform a comprehensive test analysis of GNN-based solutions, covering a wide range of network scenarios, before these tools are released to the market.

4. PRELIMINARIES

This section firstly introduces a detailed description of the representations produced by graph-based explainability methods, which are commonly referred to as *explainability masks*. Then, we present the general overview on how state-of-the-art explainability solutions produce *explainability masks* on graphs.

4.1 Explainability mask

We refer to the *explainability mask* as an $n \times n$ matrix that defines the relevance of each connection of an input graph $G = (V, E)$ on the output produced by the target GNN, where $n = |V|$. This mask enables us to interpret which are the main graph elements that affect most the predicting power of the GNN in each case.

Formally, given an input graph $G = (V, E)$, state-of-the-art explainability methods aim to produce an explainability mask $W \in \{0, 1\}^{n \times n}$, where each element defines a weight $W_{i,j}$ indicating the importance of the connection between node i and node j on the overall accuracy of the target GNN. Fig. 4 illustrates how the explainability mask is built from an input (undirected) graph G. Particularly, this matrix contains a weight for each pair (i, j) connected in the graph.

Note that when applying GNN to network-related problems, input graphs G may contain a wide variety of elements and connections that do not necessarily correspond to physical network elements (e.g., forwarding devices, links). For instance, some proposals like [12, 16] introduce complex hypergraphs including logic network entities (e.g., end-to-end paths).

4.2 Generating explainability masks

Current explainability solutions are based on iterative optimization methods, which work as follows:

Given a target GNN and an input graph $G = (V, E)$, explainability algorithms apply an iterative (costly) gradient descent method to compute the explainability mask W that best explains the accuracy of the model (i.e., it defines the set of weights W (see Eq. *(5)*)) that represents the impact of each graph edge on the loss function of the target GNN. More specifically, the calculation of the explainability mask is driven by the loss function of Eq. *(6)*, which depends on three factors: (i) predictive loss, (ii) entropy of the values in the mask, and (iii) L1 regularization computed over the mask. The predictive loss quantifies how the accuracy of the target GNN (Y_I) degrades when weighting the hidden states according to W (Y_W). Note that the predictive loss function greatly depends on the specific problem we aim to solve (e.g., regression or classification). The entropy factor (Eq. *(7)*) controls the trade-off between too homogeneous or too sparse values in the resulting mask W. Finally, the $L1$ regularization controls the number of connections that will have high values. More in detail, as the regularization factor has more importance, the mask will be driven towards having less critical connections (i.e., less high-value weights), which can be more useful for human interpretability. Moreover, notice that both entropy and regularization loss are weighted according to two hyper-parameters (i.e., α, β) that can be fine-tuned according to the problem's needs.

Through a gradient descent method, these algorithms gradually converge to the optimal mask W^* that minimizes the loss function $\ell(W)$.

$$\ell(W) = P(Y_I, V_W) + \alpha H(W) + \beta ||W||_{L1} \quad (6)$$

$$H(W) = -\sum_{i,j} (W_{i,j} \log(W_{i,j}) + (1 - W_{i,j}) \log(1 - W_{i,j})) \quad (7)$$

5. NETXPLAIN: PROPOSED EXPLAINABILITY METHOD

In this section, we introduce NetXplain, a novel explainability method for GNN, compatible with real-time operations, that addresses the performance limitations of existing solutions (Section 3). NetXplain is able to produce the same output as state-of-the-art solutions, based on costly iterative optimization algorithms [17, 3], while operating at a much limited cost (at the scale of a few milliseconds in our experiments in Section 6). This not only enables us

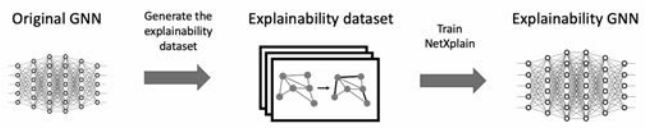


Fig. 5 – High-level workflow of NetXplain.

to perform real-time troubleshooting of GNN-based solutions applied to networks but also opens the possibility of combining these solutions with automatic optimization algorithms (e.g., local search, reinforcement learning) to solve more efficiently online optimization problems, as discussed later in Section 7. To this end, *NetXplain uses a GNN that learns how to interpret a target GNN model* that has been trained for a particular task. As shown in Fig. 5, the proposed GNN-based solution is trained with an *explainability data set* generated by an iterative optimization algorithm [3] and, once trained, the resulting model can make one-step explainability predictions for each input sample of the target GNN. Note that thanks to the generalization capabilities of GNN over graph-structured information, once NetXplain is trained over a particular target GNN solution, it can be applied to different input graphs not included in the training data set. In practice, when applied to GNN-based networking solutions, NetXplain is able to generalize to network scenarios with topologies of variable size and structure not seen in advance, as shown later in the experiments of Section 6. The following subsections describe in more detail the main components of this solution.

5.1 Explainability data set

To train NetXplain, we first need to generate the new explainability data set, which we refer to as A. To this end, we first randomly sample a subset $D' \subseteq D$, where D is the original data set used to train the target GNN. Given this subset D', we now target the problem of producing, for each input graph $G \in D'$, its associated explainability mask W_G when applied to the target GNN. Note that this process is made from a black-box perspective (i.e., the explainability mask interprets the relevance of the input graph connections by analyzing the input-output correlations in the target GNN). For this task we can use specific state-of-the-art iterative optimization algorithms, introduced in Section 3, and further described in Section 4.2, depending on the particularities and the purpose of the target GNN (e.g., regression, classification).

Thus, we apply the process described in Section 4.2 for each of the samples $G \in D'$. Hereby, we eventually obtain the final explainability data set A, formally defined in Eq. *(8)*, which maps each of the selected graphs to its corresponding optimal mask W^*_G.

$$A = \{(G, W^*_G) \mid G \in D'\} \qquad (8)$$

Note that due to the high cost of computing the explainability data set, it is crucial to ensure that $|D'| << |D|$. For instance, in our experiments, we observe that NetXplain is able to converge to a valid solution using only 5-10% of the samples of the original data sets. Consequently, the cost of generating the explainability data set becomes much more affordable than applying the iterative optimization algorithm over all the samples of D.

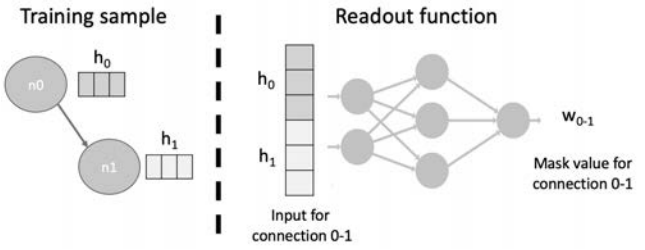


Fig. 6 – Adaptation of the readout function in NetXplain to produce the explainability mask.

5.2 Training the explainability GNN

Finally, we propose the use of an independent GNN (NetXplain) to learn how to predict explainability masks W_G over the target GNN for an input graph $G = (V, E)$.

First, we must define the underlying architecture of the NetXplain GNN, which we use for training. Particularly, we mostly keep the same architecture of the target GNN. The intuition behind this decision is that the complexity for the target GNN to learn how to make its output predictions should be similar to solving the explainability problem over that GNN (i.e., explaining which connections affected most such predictions). However, it is needed to make a minor change on the readout function $r(\cdot)$, in order to adapt it to produce the explainability mask M_G. As illustrated in Fig. 6, for every edge $(i, j) \in E$, we concatenate their final hidden-state vectors after the message passing phase is finished (i.e., $h_i^T \mid\mid h_j^T$) and this is passed as input to $r(\cdot)$, which predicts the mask weight for that edge $W_{i,j}$. Note that this operation can be computed in parallel for each node pair $(i, j) \in E$ of the input graph.

A key aspect of our proposal is to reduce as much as possible the subset of samples randomly selected (D') used to generate the samples of the explainability data set (A), which are finally used to train NetXplain's GNN. The reason is that typically producing explainability masks for all the samples of the original data set D may be too costly with state-of-the-art explainability solutions. To achieve this, we follow a transfer learning approach. Particularly, we first initialize the explainability GNN with the same internal parameters (i.e., weights and biases) of the target GNN model, except for the readout function, whose implementation differs as explained before. This enables us to effectively initialize the explainability GNN model, as the message-passing functions of this GNN are expected to be close to those of the target GNN (e.g., similar graphs and feature distributions). Thus, during training, the main adjustment should be made over the readout function. To this end, we finally train the explainability model with a reduced explainability data set A generated by a reference explainability algorithm, and this enables us to learn how to produce accurately explainability masks.

5.3 Generalization power of NetXplain

By analyzing the training process of NetXplain, we identify the generation of the data set A as the most computationally costly task, even when considering that the number of selected samples of D is only a small portion of the original data set D.

Note that our proposal aims to learn how to explain potentially any sample that our target GNN could face during operation. This motivates our choice of using a GNN to explain a target GNN. To the best of our knowledge, GNNs are the only technique that offers high generalization power over graph-structured data. As a result, once trained, the GNN explainability model generalizes to network scenarios not present in its training data set A. This means that NetXplain's GNN can be trained over a small data set to make predictions of the critical connections from the perspective of the target GNN and, once trained, it can predict in one step these critical connections over arbitrary network scenarios (e.g., topologies of variable size and structure). All this while offering an accuracy comparable to state-of-the-art costly solutions.

6. EVALUATION

In this section, we first evaluate the accuracy of the predictions made by NetXplain with respect to the state-of-the-art solutions (Metis [3]). Second, we quantify the speed-up when using NetXplain compared to Metis. In our experiments, we train an explainability model that makes interpretations over RouteNet [12], a GNN model used to make QoS inference in networks, previously introduced in more detail in Section 2.2.

All these experiments are evaluated over the same data sets used in RouteNet [12], which are publicly available at [18].

6.1 Generating the explainability model

First, we need to generate the explainability data set and define an architecture for the explainability GNN model:

6.1.1 Explainability data set

To train a NetXplain explainability model for RouteNet we first need to generate the explainability data set A (Section 5.1). In this case, we generate this data set using Metis [3].

To this end, we first train RouteNet as the target GNN model, using 300k samples simulated in the NSFNet network, including scenarios with various routing configurations and traffic matrices [18].

Before generating the explainability data set A, we randomly sample a subset $D' \subseteq D$ from the original data sets [18]. Note that the different experiments made in this section use different subsets D' to generate the explainability data sets A, finally used to train the NetXplain's GNN models. This is then specified in the respective sections. In general, our experimentation shows that this subset D' needs only $\approx 5\%$ of samples randomly extracted from the original data set D (i.e., approximately 15k samples) to ensure that NetXplain learns properly.

Afterward, we generate with Metis the final explainability data set A, as described in Section 5.2. In this process Metis maps each of the selected samples $G \in D'$ to its corresponding mask W_G, using as a target GNN the RouteNet model previously trained on samples of NSFNet. Note that Metis [3] is an iterative optimization algorithm. Hence, we limit it to run 2,000 iterations per sample, after observing this was sufficient to ensure convergence.

Finally, to train our NetXplain model, we make a random split of the explainability data set A (80%, 10%, and 10%) to produce the training, validation, and test data sets respectively.

Algorithm 1: Architecture of the NetXplain's explainability GNN applied to RouteNet

input : $\mathbf{x}_p$, $\mathbf{x}_l$,$\mathcal{R}$
output: h_v^T, h_e^T, h_p^T, W

begin
  // Initialize states of paths and links
  foreach $p \in \mathcal{R}$ **do** $h_p^0 \leftarrow [x_p, 0 \dots, 0]$;
  foreach $l \in \mathcal{N}$ **do** $h_l^0 \leftarrow [x_l, 0 \dots, 0]$;
  for $t = 1$ *to* T **do**
    // Message passing from links to paths
    foreach $p \in \mathcal{R}$ **do**
      $m_p^t = \{h_l^{t-1} \mid l \in p\}$
      $h_p^t \leftarrow RNN_p(h_p^{t-1},\ m_p^t)$
    end
    // Message passing from paths to links
    foreach $l \in \mathcal{N}$ **do**
      $m_l^t \leftarrow \sum_{p:l \in p} h_p^t$
      $h_l^t \leftarrow RNN_l\,(h_l^{t-1},\ m_l^t)$
    end
  end
  // Readout function
  foreach $p \in \mathcal{R}$ **do**
    foreach $l \in p$ **do**
      $q_{l,p} \leftarrow (h_l^T \mid h_p^T)$
      $W_{l,p} \leftarrow r(\ \mathbf{q}_{l,p}\ )$
    end
  end
end

6.1.2 Architecture of the explainability GNN

As previously mentioned in Section 5.2, we use for the explainability GNN a similar architecture to the target GNN, RouteNet [12] in this case. The only change introduced with respect to the original formulation of RouteNet is in the readout function. Algorithm 1 provides a detailed description of the NetXplain's explainability GNN when applied to RouteNet (see scheme of Fig. 2). In this case, the readout function outputs a weight $W_{l,p}$ for each link-path connection (l, p). To this end, we concatenate the corre-

sponding hidden states of the *link* (h_l) and the *path* (h_p), and introduce this as input of the readout function. Thus, the resulting weight $w_{l,p}$ can be interpreted as quantifying the importance for RouteNet of a particular src-dst path p as it passes through a certain link l of the network.

6.2 Evaluation of the accuracy

We evaluate the accuracy achieved by the NetXplain model on samples simulated in three real-world topologies [18]: *NSFNet* (14 nodes), *GEANT2* (24 nodes), and *GBN* (17 nodes). Concretely, for each topology we randomly pick 1,000 samples (with different routing configurations, and traffic matrices), and produce explainability masks with the NetXplain GNN model described in Section 6.1.2. Fig. 7 depicts the Cumulative Distri- bution Function (CDF) of the relative error produced by NetXplain's predictions with respect to those obtained by Metis [3], acting as the ground truth. We observe that our explainability model achieves a Mean Relative Error (MRE) of 2.4% when it is trained and evaluated over explainability data sets A with samples of the NSFNet topol- ogy (14 nodes). We then repeat the same experiment training and evaluating the model with samples of Geant2 (24 nodes), and obtain an MRE of 4.5%. Note that de- spite NetXplain's GNN being trained and evaluated over samples of the same topology, the network scenarios (i.e., routing and traffic matrices) are different across the train- ing and evaluation samples, which means that the input graphs seen by the GNN in the evaluation phase are dif- ferent from those observed during training. Finally, we further test the generalization capabilities of NetXplain by training the explainability GNN with samples from *NSFNet* and *GEANT2*, but in this case, we evaluate the model on samples of a different network: *GBN* (with 17 nodes). As a result, NetXplain achieves an MRE of 11%over this network topology unseen in advance (dashed line in Fig. 7). All these values are in line with the general- ization results already observed in the target GNN model (RouteNet [12]).

These results together show that using NetXplain we can achieve a similar output to a state-of-the-art solution based on iterative optimization (Metis [3]), even when our solution was tested over network scenarios not seen during training.

Table 1 – Execution time of NetXplain with respect to Metis, evaluated on three real-world network topologies

Topology	Method	Mean (s)	Std deviation (s)
NSFNet	Benchmark (Metis)	98.139	2.455
	NetXplain	0.012	0.001
GBN	Benchmark (Metis)	150.83	1.79
	NetXplain	0.0214	0.005
GEANT2	Benchmark (Metis)	191.46	2.76
	NetXplain	0.029	0.002

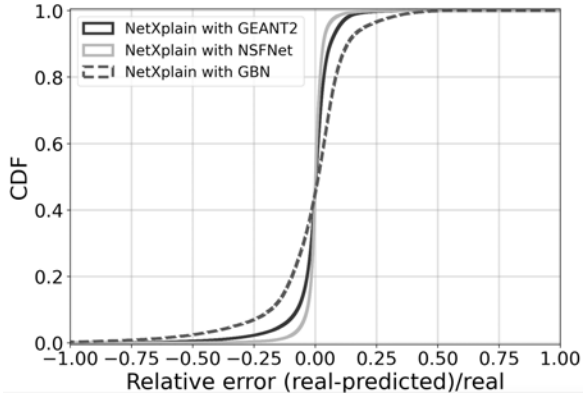


Fig. 7 – CDF of the relative error of NetXplain evaluated on three real-world network topologies.

6.3 Evaluation of the execution cost

In this section, we evaluate the computational time of NetXplain with respect to the original solution used to generate the explainability data set (Metis [3]). We thus measured the time to produce the output explainability masks using both solutions. This was done by randomly selecting 500 samples from each of the three topologies previously used in the experiments of Section 6.2: *NSFNet* (14 nodes), *GEANT2* (24 nodes), and *GBN* (17 nodes) [18]. Table 1 shows the execution times per sample during inference (in seconds), differentiated over the three considered data sets. Note that both solutions were executed in CPU and in equal conditions (they were applied over the same samples). We can observe that Metis takes ≈98 seconds on average to produce an explainability mask for an input sample of *NSFNet* (14 nodes). In contrast, NetXplain produced each mask in 12 ms on average. This constitutes a mean speed-up of ≈8,178x in the execution time. As we can observe, similar results are obtained for the samples of the other two network topologies, resulting in an average speed-up of ≈7,200x across all the topologies (i.e., more than 3 orders of magnitude faster).
This shows the benefits of NetXplain with respect to state-of-the-art solutions, as it can be used to make extensive explainability analysis at a limited cost (e.g., to delimit the safe operational range of the target GNN). More importantly, its operation at the scale of milliseconds makes it compatible with real-time networking applications.

7. DISCUSSION ON POSSIBLE APPLICATIONS

As previously mentioned, GNNs have been mainly leveraged for global network control and management tasks [3], as these scenarios typically involve modeling complex (and mutually recursive) relationships between different network elements (e.g., devices, links, paths) to then produce the system's output (e.g., end-to-end QoS metrics [12], routing decisions [15, 13]). In this section, we draw a taxonomy with three main use case categories where the application of GNN-based explainability solutions can be especially beneficial (Fig. 8):

(i) test & troubleshooting, (ii) reverse engineering, and (iii) improving optimization tasks. Particularly, we put the focus on the advantages of leveraging the fast and low-cost interpretations of NetXplain with respect to state-of-the-art explainability methods.

7.1 Test & troubleshooting

In order to achieve GNN-based products for networking, we need guarantees that they will work optimally when deployed in real-world networks. In this context, vendors would typically need to make extensive tests to their GNN solutions to check how they respond under different network conditions. Using NetXplain would enable us to collect human-readable interpretations of the internal data processing made by GNNs. For instance, if we have a GNN model that performs traffic engineering, we can identify the network elements that mainly drive the decisions made by the model, which are given by the explainability mask of NetXplain, and then observe if the properties of the selected elements are consistent across similar network scenarios. This would be a good indicator that the model generalizes well and, consequently, it is reliable for deployment. In this vein, with extensive testing we can find the safe operational range of models, which is essential for vendors to offer guarantees before selling their products (e.g., this product works optimally in networks up to 100 nodes and link capacities up to 40Gbps). Otherwise, operators would not take the risk of deploying such solutions on their networks, as they are critical infrastructures where misconfigurations are not acceptable. In this context, making such a comprehensive analysis using state-of-the-art solutions would result in large costs for vendors; while the limited cost of NetXplain would enable us to reduce dramatically both the cost and the time needed before releasing the product to the market.

Moreover, this testing process would enable us to troubleshoot GNN models by identifying particular scenarios where they are not focusing on the expected elements, or simply their behavior is not consistent with other similar scenarios. In this context, understanding *where and why* a model failed is crucial to refine it through an iterative training-testing process. For instance, it can help find deficiencies in the internal message-passing architecture that make the model less robust to particular network scenarios or identify a lack of samples in the training data sets.

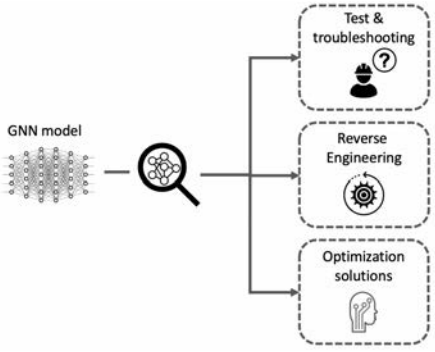


Fig. 8 – Possible applications of NetXplain.

7.2 Reverse engineering

One interesting application of ML-based solutions is to extract information about the knowledge learned during the training phase (i.e., reverse engineering). In this context, the explainability interpretations produced by NetXplain would enable us to understand what are the main network elements that GNNs consider before making their decisions. As a result, this may enable us to obtain non-trivial knowledge that can be leveraged to then design and implement efficient optimization algorithms and/or heuristics with deterministic and predictable behavior. These kinds of solutions are often perceived as more valuable by network operators, as nowadays there is a certain skepticism on applying ML-based solutions to real-world networks, mainly due to the critical nature of these infrastructures and the probabilistic guarantees typically offered by ML solutions.

7.3 Improving network optimization solutions

Network optimization problems often require dealing with very large spaces of possible actions (e.g., all the valid src-dst routing combinations in a network). As a result, optimization tools can only evaluate a small portion of configurations before they make a final decision. Thus, the exploration strategy used by these tools has a critical impact on the performance they can eventually achieve. In this context, explainability methods can provide meaningful interpretations of the current network state that can be useful to guide more efficiently optimization algorithms (e.g., reinforcement learning [15], local search [19]). For instance, using a NetXplain model trained over RouteNet, as the one of Section 6, would enable us to point to critical paths and links that are mostly affecting the network performance (e.g., end-to-end delays). This could be highly beneficial for optimization algorithms to explore alternative configurations targeting specifically these critical points (e.g., re-routing specific paths to avoid the critical points selected by NetXplain). In this context, computational efficiency is a must for optimization tools, as it directly affects the number of configurations that can be evaluated before producing the final decision. Thus, counting on solutions compatible with real-time operation, like NetXplain, offers an important competitive advantage with respect to state-of-the-art explainability solutions.

8. CONCLUSIONS

In this paper, we proposed NetXplain, an efficient explainability solution for Graph Neural Networks (GNNs). Particularly, this solution uses a GNN that *learns* how to produce accurate interpretations over the outputs produced

by another GNN model. In contrast to state-of-the-art solutions based on costly optimization algorithms, the proposed solution can be integrated into network control and troubleshooting systems operating in real time. We tested NetXplain over RouteNet, a GNN model that predicts per-source-destination delays in computer networks, and showed that our solution can produce an output equivalent to state-of-the-art solutions with an execution time more than 3 orders of magnitude faster in networks up to 24 nodes. Moreover, we discussed the potential applications that can have this GNN-based explainability solution when applied to networking. As future work, it would be interesting to show experimentally the potential applications of the proposed lightweight explainability method to different networking use cases, such as those described in Section 7, as well as making a deep analysis on the knowledge extracted by NetXplain on different target GNN models.

ACKNOWLEDGEMENT

This work has been supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE) and the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[2] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[3] Zili Meng, Minhu Wang, Jiasong Bai, Mingwei Xu, Hongzi Mao, and Hongxin Hu. "Interpreting Deep Learning-Based Networking Systems". In: *Proceedings of ACM SIGCOMM*. 2020, pp. 154–171.

[4] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models". In: *arXiv preprint arXiv:1708.08296* (2017).

[5] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Network dissection: Quantifying interpretability of deep visual representations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6541–6549.

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The graph neural network model". In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.

[7] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for quantum chemistry". In: *arXiv preprint arXiv:1704.01212* (2017).

[8] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. "Interaction networks for learning about objects, relations and physics". In: *Advances in neural information processing systems (NIPS)*. 2016, pp. 4502–4510.

[9] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. "Modeling polypharmacy side effects with graph convolutional networks". In: *Bioinformatics* 34.13 (2018), pp. i457–i466.

[10] Rex Ying et al. "Graph convolutional neural networks for web-scale recommender systems". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 974–983.

[11] Wenqi Fan et al. "Graph neural networks for social recommendation". In: *The ACM World Wide Web Conference (WWW)*. 2019, pp. 417–426.

[12] Krzysztof Rusek, José Suárez-Varela, Albert Mestres, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN". In: *Proceedings of the 2019 ACM Symposium on SDN Research*. 2019, pp. 140–151.

[13] Fabien Geyer and Georg Carle. "Learning and generating distributed routing protocols using graph-based deep learning". In: *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. 2018, pp. 40–45.

[14] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. "Learning scheduling algorithms for data processing clusters". In: *Proceedings of the ACM SIGCOMM*. 2019, pp. 270–288.

[15] Paul Almasan, José Suárez-Varela, Arnau Badia-Sampera, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case". In: *arXiv preprint arXiv:1910.07421* (2019).

[16] Arnau Badia-Sampera, José Suárez-Varela, Paul Almasan, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "Towards more realistic network models based on Graph Neural Networks". In: *Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies*. 2019, pp. 14–16.

[17] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks". In: *Advances in neural information processing systems*. 2019, pp. 9244–9255.

[18] *Network Modeling Datasets*. https://github.com/knowledgedefinednetworking/NetworkModelingDatasets. Jan. 2020.

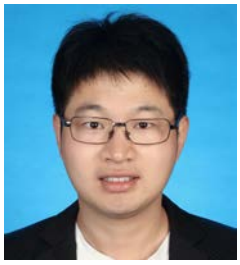
[19] Steven Gay, Renaud Hartert, and Stefano Vissicchio. "Expect the unexpected: Sub-second optimization for segment routing". In: *IEEE INFOCOM 2017*. 2017, pp. 1–9.

AUTHORS

David Pujol-Perich is an MSc student in advanced computing from UPC (2020-2022), where he is also received his Bachelor's degree in computer science (2016-2020). He is currently employed by Barcelona Neural Networking center (BNN-UPC) as a machine learning junior researcher. During his Bachelor's, he did an academic stay in ETH Zürich (2019-2020) and was awarded as finalist of the Best Bachelor Thesis of the Year.
In addition, the project resulting from his bachelor thesis, IGNNITION, received an EU grant (H2020-871528-NGI-POINTER; 1st open call). His research focuses on the development of tools that allow non-expert users in ML to easily implement complex graph neural networks. Furthermore, he is also working on novel techniques to explain the knowledge that graph neural networks automatically acquire to accurately make predictions.

José Suárez-Varela received his M.Sc. degree in telecommunication engineering from the Universidad de Granada (UGR) in 2017, and his Ph.D. from the Universitat Politècnica de Catalunya (UPC) in 2020. He is currently a post-doctoral researcher at the Barcelona Neural Networking Center (BNN-UPC), and co-Principal Investigator of the EU-funded project IGNNITION (H2020-871528-NGI-POINTER; 1st open call).
During his Ph.D., he was a visiting researcher at the University of Siena, under the supervision of Prof. Franco Scarselli. His main research interests are in the field of artificial intelligence applied to networking, and traffic monitoring and analysis.

Shihan Xiao received a B.Eng. degree in electronic and information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012, and a Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China. He is currently a senior engineer with Huawei 2012 NetLab. His research interests include machine learning in networking, data center networking, and cloud computing.



Bo Wu received his Bachelor's degree from the School of Software at Shandong University, China, in 2014, and his Ph.D. degree from the Department of Computer Science and Technology at Tsinghua University in 2019. Currently, he works in the Network Technology Laboratory at Huawei Technologies. His research interests include network architecture, network security, and blockchain.

Albert Cabellos-Aparicio is an assistant professor at Universitat Politecnica de Catalunya (UPC), where he obtained his PhD in computer science in 2008. He is director of the Barcelona Neural Networking Center (BNN-UPC) and scientific director of the NaNoNetworking Center in Catalunya. He has been a visiting researcher at Cisco Systems and Agilent Technologies and a visiting professor at KTH, Sweden, and MIT, USA. His research interests include the application of machine learning to networking and nanocommunications. His research achievements have been awarded by the Catalan Government, the University, and INTEL. He also participates regularly in standardization bodies such as IETF.

Pere Barlet-Ros is an associate professor at Universitat Politècnica de Catalunya (UPC) and scientific director of the Barcelona Neural Networking Center (BNN-UPC). From 2013 to 2018, he was co-founder and chairman of the machine learning startup Talaia Networks, which was acquired by Auvik Networks in 2018. He was a visiting researcher at Endace (New Zealand), Intel Research Cambridge (UK) and Intel Labs Berkeley (USA). His research interests are in machine learning for network management and optimization, traffic classification and network security. He received the 2nd VALORTEC prize (2014) for the best business plan awarded by the Catalan Government (ACCIO).

MACHINE LEARNING FOR PERFORMANCE PREDICTION OF CHANNEL BONDING IN NEXT-GENERATION IEEE 802.11 WLANS

Francesc Wilhelmi[1], David Góez[2], Paola Soto[3], Ramon Vallés[1], Mohammad Alfaifi[4], Abdulrahman Algunayah[4], Jorge Martín-Pérez[5], Luigi Girletti[5], Rajasekar Mohan[6], K Venkat Ramnan[6], Boris Bellalta[1]
[1]Universitat Pompeu Fabra, Spain, [2]Universidad de Antioquia, Colombia, [3]University of Antwerp, Belgium, [4]Saudi Telecom, Saudi Arabia, [5]Universidad Carlos III de Madrid, Spain, [6]PES University, India

NOTE: Corresponding author: Francesc Wilhelmi, fwilhelmi@cttc.cat

Abstract – *With the advent of Artificial Intelligence (AI)-empowered communications, industry, academia, and standardization organizations are progressing on the definition of mechanisms and procedures to address the increasing complexity of future 5G and beyond communications. In this context, the International Telecommunication Union (ITU) organized the first AI for 5G Challenge to bring industry and academia together to introduce and solve representative problems related to the application of Machine Learning (ML) to networks. In this paper, we present the results gathered from Problem Statement 13 (PS-013), organized by Universitat Pompeu Fabra (UPF), whose primary goal was predicting the performance of next-generation Wireless Local Area Networks (WLANs) applying Channel Bonding (CB) techniques. In particular, we provide an overview of the ML models proposed by participants (including artificial neural networks, graph neural networks, random forest regression, and gradient boosting) and analyze their performance on an open data set generated using the IEEE 802.11ax-oriented Komondor network simulator. The accuracy achieved by the proposed methods demonstrates the suitability of ML for predicting the performance of WLANs. Moreover, we discuss the importance of abstracting WLAN interactions to achieve better results, and we argue that there is certainly room for improvement in throughput prediction through ML.*

Keywords – Channel bonding, IEEE 802.11 WLAN, ITU Challenge, machine learning, network simulator

1. INTRODUCTION

The utilization of Artificial Intelligence (AI) and Machine Learning (ML) techniques is gaining momentum to address the challenges posed by next-generation wireless communications. The fact is that networks are nowadays facing unprecedented levels of complexity due to novel use cases including features such as spatial multiplexing, multi-array antenna technologies, or millimeter wave (mmWave) communications. While these features allow providing the promised performance requirements in terms of data rate, latency, or energy efficiency, their implementation entails additional complexity (especially for crowded and highly dynamic deployments), thus making hand-crafted solutions unfeasible.

IEEE 802.11 Wireless Local Area Networks (WLANs) are one of the most popular access solutions in the unlicensed band, and they represent a prominent example of increasing complexity in wireless networks. The optimization of WLANs underlines particular challenges due to the decentralized nature of these types of networks, which mostly operate using Listen-Before Talk (LBT) transmission procedures. If to this we add that WLAN deployments are typically unplanned, dense, and highly dynamic, the complexity is even increased.

To address the optimization of next-generation WLANs, the usage of AI/ML emerges as a compelling solution by leveraging useful information obtained across data, which allows deriving models from experience. Nevertheless, the adoption of AI/ML in networks is still in its initial phase, and a lot of work needs to be done. In this regard, standardization organizations are undertaking significant efforts towards fully intelligent networks. An outstanding example can be found in the International Telecommunication Union (ITU)'s ML-aware architecture [1], which lays the foundations of pervasive ML for networks.

Another aspect essential for the prosperity of AI/ML in communications is data availability and openness. In this context, the *ITU AI/ML in 5G Challenge* [2] was set in motion to encourage industry, academia, and other stakeholders to collaborate and exchange data for solving relevant problems in the field. This initiative entailed a big step forward in bringing open source closer to standards.

As a contribution to the ITU challenge, and aligned with WLANs optimization, in this paper, we present the results obtained from problem statement *"Improving the capacity of IEEE 802.11 Wireless Local Area Networks (WLANs) through Machine Learning"* (referred to as PS-013 in the context of the challenge), whereby participants were called to design ML models to predict the performance of next-generation Wi-Fi deployments. In this article, we gather all the work done in the context of the above-mentioned problem statement and provide a compilation of the proposed ML models used to address the problem of CB in WLANs.

To carry out the ITU AI for 5G challenge, an open data set with WLAN measurements obtained from a network simulator was made available. As later discussed, network simulators are gaining importance to enable future ML-aware communications by acting as ML sandboxes. In the context of the challenge, synthetic data was used for training ML models.

In summary, based on the proposed problem statement and the solutions provided by participants, we discuss the feasibility of predicting the throughput of complex WLAN deployments through ML. To the best of our knowledge, this is an under-researched subject with high potential. Accurate performance predictions may open the door to novel real-time self-adaptive mechanisms able to enhance the performance of wireless networks by leveraging spectrum resources dynamically. For instance, given a change in the network configuration, predictions about future performance values can be used as a heuristic to guide the choices made by algorithms that operate in decentralized self-configuring environments [3].

Table 1 briefly summarizes all the models proposed by the participants of the challenge and the main motivation behind them. For instance, the model proposed by the *ATARI* team aims to exploit the graph structure inherent in WLAN deployments. Alternatively, the solution provided by *Ramón Vallés* is focused on abstracting different categories of features (e.g., signal quality, bandwidth usage) and generate predictions based on them.

Table 1 – Summary of the ML models proposed by the participants of the challenge.

Team	Proposed Model	Motivation	Ref.
ATARI	Graph Neural Network	Exploit graph representation of WLANs	[29]
Ramon Vallés	Feed-forward Neural Network	Abstract problem characteristics by categories	[31]
STC	Gradient Boosting	High performance, flexibility, and ease of deployment	[35]
UC3M NETCOM	Feed-forward Neural Network	Learn throughput function exhaustively	[36]
NET INTELS	Random Forest Regression	Address problem's non-linearity and reduce dimensionality	[37]

The results presented in this paper showcase the feasibility of applying ML to predict the performance of next-generation WLANs. In particular, some of the proposed models have been shown to achieve high prediction accuracy in a set of test scenarios. Moreover, we have identified the main potential and pitfalls of the proposed models, thus opening the door to new contributions that improve the baseline results shown in this paper. The data set is available online [4], and we expect it can be used for benchmarking other ML methods in the future.

The remainder of this paper is structured as follows: first, Section 2 presents the CB problem for next-generation WLANs and describes the data set provided for throughput prediction in dense deployments. Then, Section 3 gives an overview of the ML-based solutions proposed by the ITU challenge participants, for which the results are provided in Section 4. Finally, Section 5 concludes the paper with final remarks.

2. CHANNEL BONDING IN NEXT-GENERATION WLANS

In this section, we first describe the CB problem in WLANs and underscore the need for ML models for performance prediction. Then, we introduce the data set provided for the ITU challenge, which is open to any researcher interested in this topic.

2.1 Channel bonding in IEEE 802.11 WLANs

Next-generation IEEE 802.11 WLANs are called to face the challenge of providing high performance under complex situations, e.g., to provide high throughput in massively crowded deployments where multiple devices coexist within the same area. To fulfill the strict requirements derived from novel use cases, features such as Multiple-Input Multiple-Output (MIMO), Spatial Reuse (SR), or multi-Access Point (AP) coordination are being developed and incorporated into the newest amendments, namely IEEE 802.11ax (11ax) and IEEE 802.11be (11be) [5, 6].

One of the features that are receiving more attention is Channel Bonding (CB) [7, 8], whereby multiple frequency channels can be bonded with the aim of increasing the bandwidth of a given transmission, thus potentially improving the throughput. Since its introduction to the 802.11n amendment, where up to two basic channels of 20 MHz could be bonded to form a single one of 40 MHz, the specification on CB has evolved and currently allows for channel widths of 160 MHz (11ac/11ax). Moreover, CB is expected to support up to 320 MHz channels in the 11be amendment.

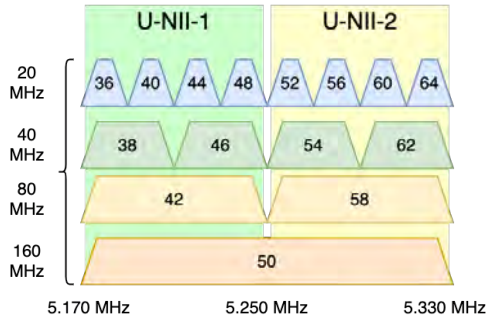


Fig. 1 – U-NII-1 and U-NII-2 sub-bands of the 5 GHz Wi-Fi channels.

Fig. 1 shows a snapshot of the 5 GHz band in Wi-Fi (particularly, U-NII-1 and U-NII-2 bands are shown), where basic 20 MHz channels can be bonded to form wider channels of up to 160 MHz. Each allowed channel is represented by an identifier. For instance, channel 36 is the first 20 MHz channel in the U-NII-1 band. As can be appreciated, the number of combinations for bonding channels is high, even for a small portion of the available spectrum[1] and under the constraint that only contiguous channels can be bonded. Moreover, novel bonding techniques combine Orthogonal Frequency Multiple Access (OFDMA) with preamble puncturing [9] to use non-contiguous channels. With all this, given the number of Basic Service Sets (BSSs) and devices in crowded deployments (see Fig. 2), we can say that CB is a problem with a combinatorial action space.

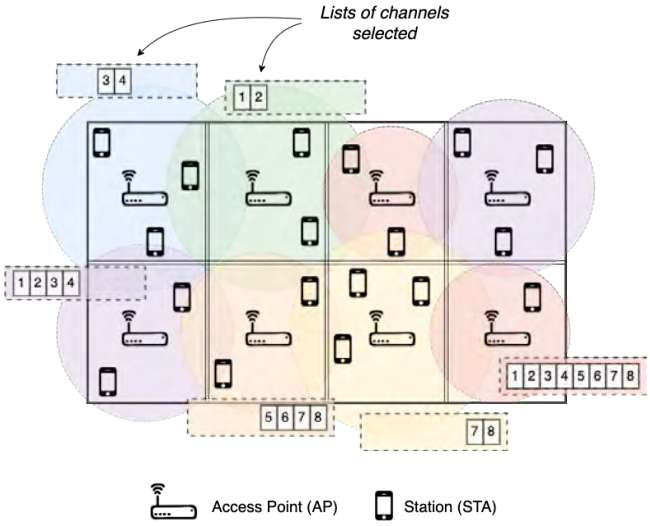


Fig. 2 – Dense WLAN deployment with multiple CB configurations. Each number in the list of channels represents one basic 20 MHz channel.

2.2 Policies for dynamic channel bonding

To harness the available spectrum within the CB operation, Dynamic CB (DCB) mechanisms [7, 8] are applied to decide the set of channels for transmitting on a per-packet basis, thus potentially improving performance. In [8], the following DCB policies were proposed and analyzed:

- **Static Channel Bonding (SCB):** a transmitter is allowed to use the entire set of channels only, thereby limiting the election of any subset of channels. While such a policy may optimize the performance in isolated deployments, it lacks the necessary flexibility to deal with inter-BSS interference.
- **Always-Max (AM):** in this case, the widest combination of channels is picked upon having sensed them free during the back-off procedure. While such a policy seems to properly harness the available spectrum, it has also been shown to generate starvation and other issues as a result of inter-BSS interactions.
- **Probabilistic Uniform (PU):** as an alternative to SCB and AM, PU is introduced to add some randomness in the process of picking free channels, so that any combination is chosen with the same probability. This policy has been shown to improve both SCB and AM in some scenarios in which flow starvation was present due to inter-BSS interactions.

To better illustrate the behavior of CB policies, Fig. 3 shows a simplification of the transmission procedure that a Station (STA) follows when implementing AM. In particular, CB may be applied over channels 1-4. Based on the AM policy, at time t_1, the STA can transmit only over channel 2, which is the only one that is sensed free at the moment of initiating a transmission. Similarly, in t_2, both channels 1 and 2 are found free, so the transmission is performed over those two channels. Finally, provided that the entire spectrum is free, a transmission over channels 1-4 is performed at t_3. Notice that, if applying SCB, the STA would have not been able to transmit until t_3. Alternatively, regarding PU, any combination of free channels could have been selected in t_2 and t_3.

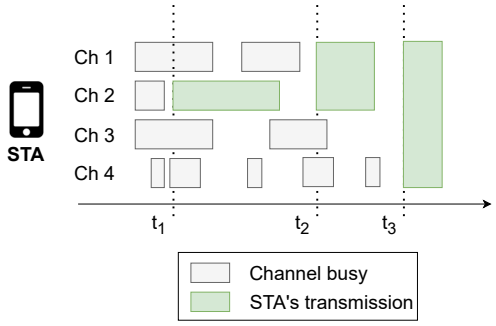


Fig. 3 – Dynamic channel selection for transmitting when applying DCB.

Through the analysis conducted in [8], it was shown that the right channel choice is not always trivial (i.e., selecting the widest channel does not necessarily entail achieving the highest performance). First of all, using wider channels entails spreading the same transmit power over the selected channel width, which can potentially affect the data rate used for the transmission, and therefore the capabilities of the receiver on decoding data successfully. Moreover, the potential gains of DCB in crowded deployments are hindered by the interactions between Wi-Fi devices, which may provoke contention or collisions. The fact is that WLAN deployments are unplanned and operate under Carrier Sense Multiple Access (CSMA). From the perspective of a given transmitter-receiver pair, such a lack of coordination leads to uncontrolled interference that can potentially degrade their performance.

Other DCB mechanisms were proposed in [10, 11, 12, 13], which include collision-detection, carrier sensing adaptation, or traffic load awareness. More recently, ML and game theory have been applied to address CB as an online decision-making problem involving multiple agents [14, 15].

[1] In the 5 GHz and 6 GHz bands, there are six and fourteen non-overlapping channels of 80 MHz, respectively.

In view of the complexity of selecting the best configuration of channels, the proposed problem statement had the goal of shedding light on the potential role of ML in DCB. In particular, it served to gather participants' proposals of ML models able to predict the performance of different CB configurations. This information can be used by a decision-making agent to choose the best configuration of channels before initiating a transmission. Throughput prediction in WLANs has been widely adopted for performance analysis through mathematical models, including the well-known Bianchi model [16], Continuous-Time Markov Networks (CTMNs) [17], or stochastic geometry [18]. However, these well-known models lack applicability for online decision-making because they fail to capture important phenomena either on the PHY or the MAC, or they entail a high computational cost. Thus, modeling high-density complex deployments through these models may be highly inaccurate or simply intractable.

In this regard, we envision ML models to assist the CB decision-making procedure in real time. The fact is that ML can exploit complex characteristics from data, thus allowing to solve problems that are hard to solve by hand-programming (see, for instance, its success in image recognition). Moreover, ML models can be trained offline and then improve their accuracy with measurements acquired online. To the best of our knowledge, this is an under-researched subject. While ML has been applied for predicting aspects related to Wi-Fi networks, such as traffic and location prediction [19, 20], it has been barely applied for explicitly predicting their performance. In this context, the work in [21] provided an ML-based framework for Wi-Fi operation, which includes the application of Deep Learning (DL) for waveforms classification, so that WLAN devices can identify the medium as idle, busy, or jamming. Closer in spirit to our work, [22] proposed an ML-based framework for WLANs' performance prediction.

2.3 Introduction to the data set

To motivate the usage of ML for predicting WLANs' performance, we provide an open data set[2] obtained with the Komondor simulator.[3] Komondor is an open-source IEEE 802.11ax-oriented simulator, whose fundamental operation has been validated against ns-3 in [23]. Komondor was conceived to cost-effectively simulate complex next-generation deployments implementing features such as channel bonding or spatial reuse [24]. Furthermore, it includes ML agents, which allows simulating the behavior of online learning mechanisms to optimize the operation of WLANs during the simulation.

[2]The data set has been made publicly available at https://zenodo.org/record/4059189, for the sake of openness.
[3]https://github.com/wn-upf/Komondor, Commit: d330ed9.

The data set generated with Komondor has been used for training and validating ML models in the context of the ITU AI for 5G Challenge. The assets provided comprise both training and test data sets corresponding to multiple random WLAN deployments at which different CB configurations are applied. As for training, two separate enterprise-like scenarios, namely, *training1* and *training2*, have been characterized. In each case, a different fixed number of BSSs coexist in the same area, according to users' density.

In *training1*, there are 12 APs, each one with 10 to 20 associated STAs. Regarding *training2*, it contains 8 APs with 5 to 10 STAs associated with each one. For both training scenarios, three different map sizes have been considered (*a*, *b*, and *c*), where STAs are placed randomly. Similarly to training scenarios, the test data set includes a set of random deployments depicting multiple CB configurations and network densities. In this case, four different scenarios have been considered according to the number of APs (4, 6, 8, and 10 APs). Note, as well, that, for each type of scenario, 100 and 50 random deployments have been generated for training and testing, respectively. In all the cases, downlink UDP traffic was generated in a full-buffer manner (i.e., each transmitter always has packets to be delivered). Table 2 summarizes the entire data set in terms of the simulated deployments.

Table 2 – Summary of the simulated deployments used for generating both training and test data sets.

	Scenario id	Map width	# APs	# STAs
Training	training1a	80 x 60 m	12	10-20
	training1b	70 x 50 m		
	training1c	60 x 40 m		
	training2a	60 x 40 m	8	5-10
	training2b	50 x 30 m		
	training2c	40 x 20 m		
Test	test1	80 x 60 m	4	2-10
	test2		6	
	test3		8	
	test4		10	

With respect to input features, these are included in the files used for simulating each random deployment. In particular, the most relevant information to be used for training ML models is:

1. **Type of node:** indicates whether the node is an AP or an STA.
2. **BSS id:** identifier of the BSS to which the node belongs.
3. **Node location:** {x,y,z} coordinates indicating the position of the node in the map.
4. **Primary channel:** channel at which carrier sensing is performed.

5. **Channels range:** minimum and maximum channels allowed for bonding.
6. **Transmit power:** power used for transmitting frames.
7. **Sensitivity threshold:** used for detecting other transmissions and assess the channels' availability.
8. **Received Signal Strength Indicator (RSSI):** power detected at receivers from their corresponding transmitters.
9. **Inter-BSS interference:** power sensed from other ongoing transmissions.
10. **Signal-to-Interference-plus-Noise Ratio (SINR):** average SINR experienced during packet receptions.

Regarding output labels, we provide the throughput obtained by each device during the simulation, being the APs' throughput the aggregate throughput of each BSS (i.e., the sum of all the individual STAs' throughput in a given BSS). Moreover, the airtime per AP is provided, which indicates the percentage of time each BSS has occupied each of its assigned channels.

3. MACHINE LEARNING SOLUTIONS FOR THROUGHPUT PREDICTION

In this section, we give an overview of the solutions proposed by the participating teams of PS-013 in ITU AI for 5G Challenge: *ATARI* (University of Antwerp and Universidad de Antioquia), *Ramon Vallés* (Universitat Pompeu Fabra), *STC* (Saudi Telecom), *UC3M NETCOM* (Universidad Carlos III de Madrid), and *Net Intels* (PES University). From these teams, ATARI, Ramon Vallés, and STC succeeded to advance to the Grand Finale, where teams from all the problem statements competed for winning the global challenge [25].

3.1 ATARI

Wireless networks can be represented by graphs G=(V, E), where V is the set of nodes, i.e., STAs and APs, and E represent wireless links. Typically, DL approaches deal with graph-structured data by processing the data into simpler structures, e.g., vectors. However, nodes and links in high-density WLAN deployments are characterized by a set of high-dimensional features, thus complicating the graph-like structure of the problem and therefore hindering the application of deep learning.

To overcome the problem of data representation, Graph Neural Networks (GNNs) have been proposed as neural networks that operate on graphs intending to achieve relational reasoning and combinatorial generalization [26]. Accordingly, the wireless interactions between STAs and APs (connectivity, interference, among others) can be easily captured via a graph representation.

Therefore, we select a GNN approach to predict the throughput of the devices in a WLAN. In particular, each deployment is considered as a directed graph where STAs and APs are the graph's nodes. Additionally, we define two types of nodes present in the data set, each one having generic or specific features. For instance, parameters like channel configuration are related to both types of nodes, while SINR is only related to STAs, and airtime is exclusive for APs. Furthermore, the edges are defined based on the type of wireless interaction derived from the data set. We consider two types of interactions, namely AP-AP interactions (represented by the interference map), and AP-STA interactions (represented by the RSSI values). For completeness, we define an additional edge feature based on the distance of every transmitter-receiver pair. The features considered for training the proposed GNN are summarized in Table 3.

Table 3 – Features used by ATARI team to train a GNN.

	Feature	Preprocessing
Node	Node type	AP=0, STA=1
	Position (x,y)	None
	Primary channel	Combined into a categorical variable using one-hot encoding
	Min. channel	
	Max. channel	
	SINR	None
	Airtime	Mean
Edge	Edge type	AP-AP=0, AP-STA=1
	Distance	Computed from (x,y)
	RSSI	None
	Interference	None

Concerning the GNN model, we have used an implementation of the Graph Network Block (GNB), as proposed in [27]. A GNB contains three update functions and three aggregation functions where the computation is done from the edge to nodes, and then to global parameters. So first, the edge's features are updated and aggregated into the node features, then the node features are updated having taken into account the vicinity within depth/range defined by the number of GNBs, and lastly, the global parameters are updated according to the state of the nodes. Our model follows a layered approach, similar to DL, where each layer is a GNB. The input of the model is a graph representing the deployment, and the output is the predicted throughput of the devices in that deployment. A general overview of our model's architecture is shown in Fig. 4.

The implementation of the GNB is referenced as a meta-layer in PyTorch Geometric [28], a geometric deep learning extension library for PyTorch. We defined an edge model that uses two dense layers using a Rectified Linear Unit (ReLU) as an activation function in a typical Multi-Layer Perceptron (MLP) configuration. A node model is also defined by two MLPs, one for aggregating the edge features into the node features and the second to update

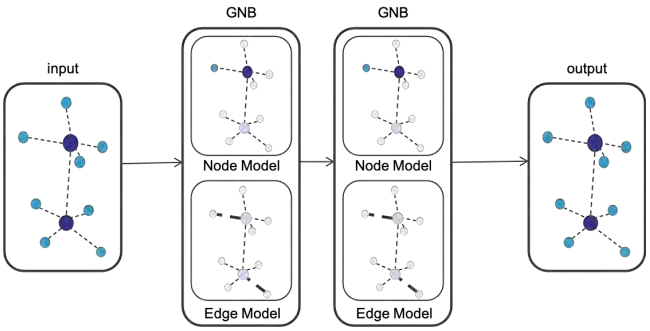


Fig. 4 – GNN model proposed by ATARI.

nodes' states based on their neighbors. Note that, following the formal definition of a GNB in [27], global parameters were not used. Our implementation is available in GitHub [29].

To train the model, we have considered splitting the provided data set (80% for training and 20% for validation). As each deployment is considered to be represented by a graph, 480 graphs have been used for training, and 120 for validation purposes. The loss function considered to assess the performance of our model is the Root Mean Squared Error (RMSE). The error obtained across all the predictions is used to compare the accuracy of our model's predictions to the actual results. However, initial results showed that the model mostly focused on predicting the throughput of the APs, given that the error is minimized on large values. Therefore, we proposed a masked loss provided that AP's throughput should be equal to the sum of the associated STAs' throughput. The RMSE is calculated using the STAs' predicted and the APs' computed throughput.

3.2 Ramon Vallés

To address the throughput prediction problem in CB-compliant WLANs, we propose a deep neural network where the information of each BSS is processed independently, thus following the idea of Multi-Layer Perceptron (MLP) [30]. More specifically, the proposed model is a feed-forward deep learning algorithm implemented in Python with the support of the PyTorch libraries.[4]

Our model aims to predict the aggregate throughput of each BSS, rather than the individual throughput at STAs. The fact is that predicting the throughput per STA is very challenging because of the dynamic channel bonding policy used in complex scenarios, which contributes to generating multiple interactions among nodes that cannot be captured at a glance. Accordingly, to derive an overall representation of each BSS, the features from individual STAs are preprocessed so that we consider only their global distribution (mean, and standard deviation). To do so, our model is divided into three main blocks performing different tasks:

1. **Signal quality:** The first block is meant to abstract the interactions among APs and STAs in the RF domain. To achieve this, we consider two separate layers, which process the RSSI, SINR, distance among nodes, and SINR. A Parametric Rectified Linear Unit (PReLU) activation function is used together with 1-dimensional batch normalization.
2. **AP bandwidth:** The second block analyzes the available bandwidth of the APs, and it consists of a single linear layer, which receives as input a vector with the corresponding airtime for all the available channels. This layer outputs a 3-dimensional array and is activated with a PReLU function.
3. **Output:** Finally, the last block takes the output from both the signal quality and AP bandwidth blocks and computes the final prediction value. For this final layer, we employ a simple ReLU (instead of a PReLU) to avoid negative throughput predictions.

With this structure, we have built a much more efficient model than if we had used a fully-connected NN with all the STA features. Notice that the proposed model needs far fewer neurons (and thus, less computational force) to capture the most relevant information of each scenario. In our opinion, an excess of neurons in such a complex scenario would result in overfitting, thus making the model less accurate for predicting the performance of new deployments.

As for training the MLP model, we have considered the following key features: the type of device (AP or STA), its location, the minimum, and maximum channel through which it is allowed to transmit, the RSSI, the SINR, and the airtime. The training was performed using 80% of the data set (keeping the 20% left for validation), following an evaluation criterion based on the RMSE of the predicted throughput with respect to the real value. The Adam optimizer has been used to optimize the training process, which is straightforward to implement, computationally efficient, and memory-efficient. For the training phase, several experiments had been made by modulating the hyper-parameters. The best results were achieved with a learning rate of 0,025 and a total of 700 epochs.

3.3 STC

Our proposal includes popular ML regression algorithms such as MLP, Support Vector Machine (SVM), Random Forest, and eXtreme Gradient Boosting (XGboost). These algorithms are backed by rich research, known to do well on regression problems, ease of implementation and deployment, which are important characteristics from the business perspective.

[4]The code used to implement the method proposed by Ramon Vallés is open access [31].

From all the above-mentioned methods, XGBoost [32] was selected for competing in the challenge because it offered the highest performance on both the training and validation stages compared with the other models. XGboost is a gradient boosting framework available for multiple platforms, thus providing high portability.

To train our model, we have analyzed the variance of the variables in the data set, and discarded the features with low or moderate variability. Some of the considered features are the position of nodes, the primary channel, the distance among nodes, or the power received by neighboring devices. To preprocess the selected features, we have applied Yeo-Johnson transformation [33] and normalization. These steps were done for predictors to improve their utility in the models.

The training data set was split into training and validation, so that we could train the model on the whole training data set using a 10-fold cross-validation procedure. Further, we used 100 and 300 combinations, respectively, with 10-fold cross-validation.

As for the hyper-parameter setting (e.g., max depth or minimum child weight), we tuned hyper-parameters using a grid search. More specifically, the hyper-parameter values were set using Latin Hypercube Sampling, while their maximum and lower range value for each hyper-parameter were mostly predetermined using default values from tidymodels [34].

Finally, significant efforts have been put to deploy our model. In particular, we have used docker to make our model easy to (re)train and deploy. All the code and documentation has been made publicly available [35].

3.4 UC3M NETCOM

We formulate the throughput forecasting in WLANs as a linear regression problem, which assumes a linear relationship between the features and label(s) of a given data set $\{y_i, x_{i,1}, x_{i,2}, ..., x_{i,N}\}$, and a set of unknown parameters w to be learned (being w_0 the bias).

Our solution (named *Gossip*) is based on a linear regression method, and it aims to predict the throughput of an STA in a given WLAN deployment where CB is applied [36]. Based on STAs' individual throughput, we derive the performance of each AP by aggregating the values of their associated STAs. In particular, Gossip derives the unknown bias and weight parameters w by (i) processing the WLAN data set, and (ii) applying a neural network to perform regression.

As for the processing part, Gossip takes the input features generated by the Komondor simulator, and selects/generates the most relevant ones: the position of the STA, the AP to which the STA is associated, the RSSI, the SINR, the set of nodes using the same primary channel, and the set of allowed channels. After processing the features, each STA of every deployment is characterized by a feature vector (x_i, ...,x_{21+3k}), with k denoting the number of wireless channels. Note, as well, that the entire data set is considered for training, thus combining STAs from different deployments. The rationale is that features such as the number of neighbors in the primary channel, the SINR, and the interference should differentiate STAs from different deployments.

When it comes to the regression problem, Gossip uses a feed-forward neural network with four layers. The input layers pass the input features to two fully connected layers of neurons with a ReLU activation unit. Finally, a single neuron receives the output of the hidden layers and generates the prediction of the throughput. As a remark, the last neuron has a linear activation. It is important to remark that the proposed neural network is mostly meant to tackle linear regression problems. Nevertheless, even if the throughput prediction problem for WLANs is not linear, we expect our model to properly identify local minimum/maximum points that allow providing reasonable prediction results.

To train the proposed neural network, we have used the RMSprop gradient descend method, considering the Mean Squared Error (MSE) as a loss function. Moreover, 50 training episodes and a batch size of 50 STAs have been considered. Thanks to Gossip design, the training data set is populated with every STA of every deployment present among all scenarios.

3.5 Net Intels

To address the objective of predicting the throughput of APs and STAs in typical dense environments, we explore a set of popular regression techniques. With the help of these techniques, we aim to build complex mathematical relationships among features and labels from the data set, so that performance of WLANs can be predicted at unseen deployments. In particular, we propose using the following techniques:[5]

1. **Artificial Neural Network (ANN):** The ANN method is selected chiefly due to its potential and versatility to model nonlinear and complex relationships in OBSS data elegantly. The proposed ANN is built using Tensorflow and Keras libraries in Python [38]. The NN model is designed with one input layer, 7 hidden layers, and 1 output layer (see Fig. 5). The ReLU function is employed to activate hidden layers. In each of the first six hidden layers, there are 1024 nodes. For the seventh hidden layer, there are 512 nodes. The model is trained using an Adam optimizer. The batch size and number of epochs for training, after multiple trials, were set to 250 and 1000 respectively.

[5] The code used to implement all the proposed methods by Net Intels is available in Github [37].

2. **K-Nearest Neighbor (KNN) regression:** For OBSSs involving several AP-STA combinations, the dynamics of interrelations between entities of OBSS rely predominantly on the relative positioning of AP/STAs. To abstract such complexity in a cost-effective manner, KNN is selected, which is characterized by its simplicity, speed, and protection against high variance and bias. The KNN model is built using the Scikit Learn library in Python [39]. The inbuilt *KNearestRegressor* function is directly used, where neighbor number is fixed to 10. The algorithm for structuring the k-dimensional space of the data set (Ball Tree, KDTree, or Brute Force) is automatically selected based on input values.

3. **Random forest regression:** Motivated by the fact that the interrelationship between the features is non-linear, we propose dividing the data set dimensional space into smaller subspaces. To generalize the data and for better feature importance, an ensemble of trees forming a random forest is used. Random forest mechanisms are useful to reduce the spread and diversion of predictions. The proposed random forest regression is built using Scikit Learn, an ensemble module of the Sklearn library. The default number of trees was set to 100, which split is performed according to the mean squared error function. The maximum depth of the tree is set to 10.

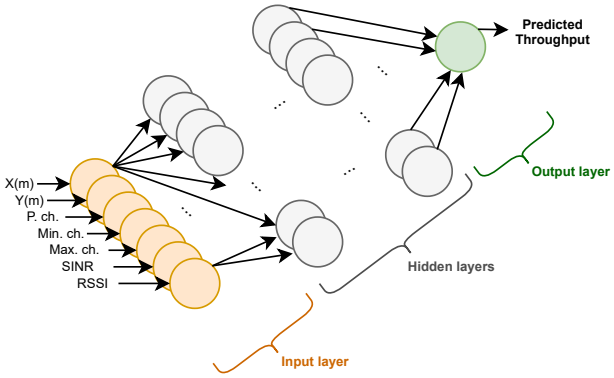


Fig. 5 – Net Intels' ANN architecture.

For all the proposed methods, we have first preprocessed the data set comprising six hundred different random deployments. In particular, static features such as the Contention Window (CW) were not included for training purposes. As for the rest of the features, we noticed a low correlation degree (see Fig. 6), so we have used all the features with higher variability from one simulation to another, including the node type (used when considering both APs and STAs during training), X and Y coordinates, primary channel, minimum and maximum channel allowed, SINR, and RSSI values.

The data was normalized before being fed into the regression models. Only data for STAs (stations) are considered for training and the throughput values of STAs are predicted using the models. The sum of the throughputs of

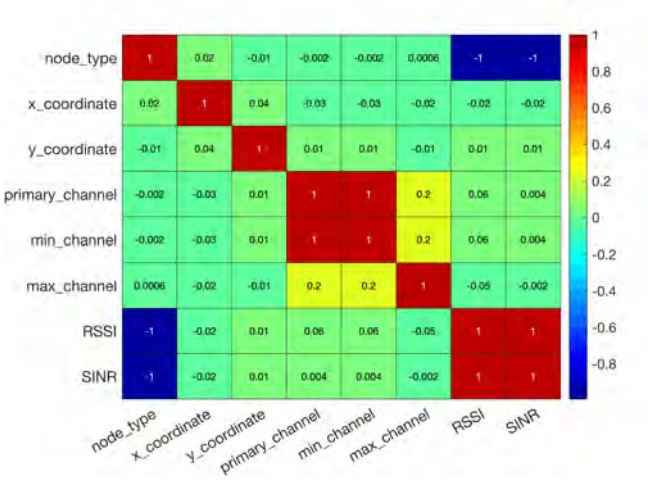


Fig. 6 – Correlation among input features.

STAs gives the throughput of the respective AP. For training purposes of all the three methods, the data is split (80% for training and 20% for validation).

4. PERFORMANCE EVALUATION

In this section, we show the results obtained by the participants' models presented in Section 3. In the context of the ITU AI for 5G Challenge, a test data set was released to assess the performance of each model, without revealing the actual throughput obtained through simulations. Participants were asked to predict the performance in Mbps of each BSS in the test scenarios.

The test data set consists of random deployments with different characteristics than the ones provided in the training data set, ranging from low to high density in terms of the number of BSSs and users. In total, test scenarios consist of 200 random deployments containing 1.400 BSSs and up to 8.431 STAs (randomly generated). To assess the participants' model accuracy, we focused on the throughput of the BSSs in each deployment (i.e., the throughput of each AP). Specifically, we used both the RMSE and the Mean Absolute Error (MAE) as reference performance metrics. Accordingly, Fig. 7 shows the MAE in Mbps obtained by each team in each type of test scenario.

As shown, for the aggregate BSS performance, most of the models offer low accuracy for the less dense scenarios (namely, *test1* and *test2*), whereas higher accuracy is achieved for the densest deployments (namely, *test3* and *test4*). The fact is that denser deployments are much more similar to the training scenarios than the sparser ones. As a result, models behave pessimistically in low-density deployments by assuming lower performance even if interference is low. As an exception, we find the model provided by Ramon Vallés, a feed-forward neural network with three blocks. The main difference of this model with respect to the others is that it separates the features related to signal quality and interference, and processes

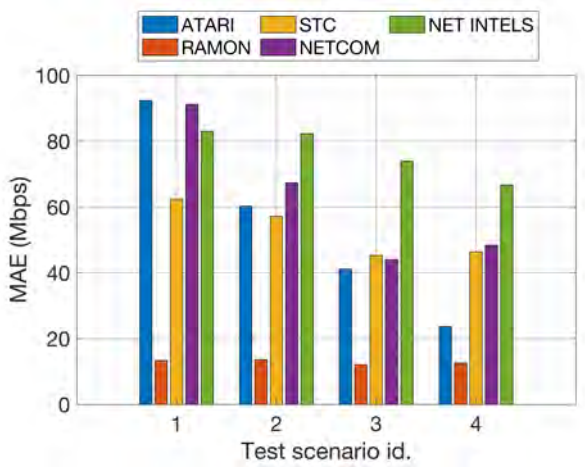


Fig. 7 – Mean absolute error obtained by each team, for each of the test scenarios of the data set. The aggregate throughput of all the BSSs is considered.

them apart from the rest. As a result, it is able to generalize well, even for new deployments with characteristics unseen in the training phase.

Although the prediction error is high for some test scenarios, it is important to remark that the performance of WLANs applying CB can be up to a few hundreds of Mbps (especially in sparse scenarios with low competition). To better illustrate the accuracy of the proposed models, we now show the prediction results obtained on a per-STA basis. Notice that the following results correspond to the solutions provided by three teams (*ATARI*, *STC*, and *Net Intels*), whose solution was based on predicting the throughput of STAs, and providing the aggregate performance afterward. Note, as well, that the target of the challenge was predicting the aggregate throughput in each BSS. In particular, Fig. 8 shows the histogram of the individual throughput predictions at STAs obtained across all the random test deployments.

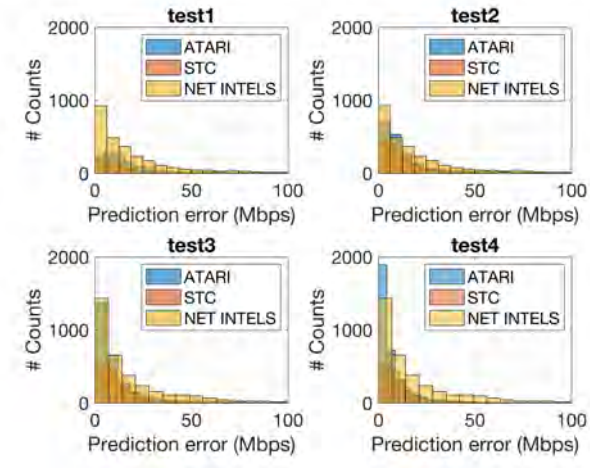


Fig. 8 – Histogram of the per-STA prediction error achieved by ATARI, STC, and NET INTELS.

As shown, the proposed ML mechanisms provide general accurate predictions; most of the error values are in the range of 0 to 10 Mbps. This means that, even in the presence of outliers, the predictions provided by the ML models are suitable for a significant percentage of the deployments. The accuracy of the different models proposed by ATARI, STC, and NET INTELS can be further observed in Table 4, which shows the percentage of the throughput predictions for STAs achieving an error below 10 Mbps.

Table 4 – Percentage of per-STA predictions achieving <10 Mbps error. Information is provided for ATARI, STC, and NET INTELS results.

	test1	test2	test3	test4
ATARI	36.97%	55.81%	67.01%	77.40%
STC	55.97%	56.27%	56.74%	60.67%
NET INTELS	38.09%	42.15%	44.01%	49.77%

For completeness, Fig. 9 shows the actual throughput achieved by STAs in all the test scenarios. As shown, the median is around 20 Mbps, but maximum values of up to 40 Mbps are also likely. Furthermore, several outliers were noticed, leading to up to 50 Mbps in some STAs.

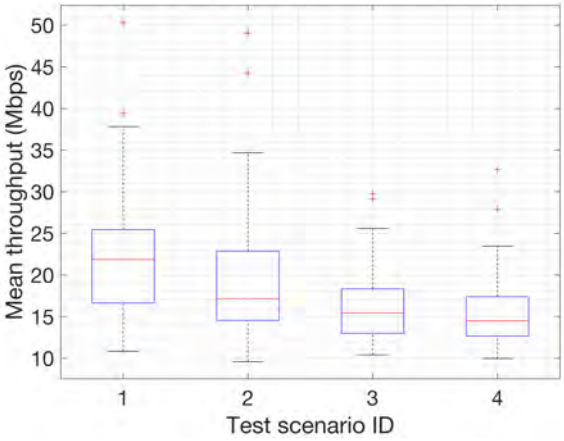


Fig. 9 – Boxplot of the mean throughput achieved by STAs for each test scenario.

Finally, to provide some insight on the computational needs required by the types of ML methods discussed in this paper, Table 5 contains the time required for training each model used by the team *Net Intels*, as well as the amount of computational resources employed. As shown, the training times are acceptable for providing near-real-time solutions.

Table 5 – Training time and computational resources used by the ML models proposed by *Net Intels*.

	Training time	RAM/GPU used
ANN	349 s	8 Gb RAM / 1.3 Gb GPU
Random forest	69 s	55 Gb RAM/ No GPU usage
KNN	122 s	5.3 Gb RAM/ No GPU usage

5. DISCUSSION

5.1 Contributions

With contributions from participants around the globe, the ITU AI for 5G Challenge has unprecedentedly established a platform for addressing important problems in communications through ML. As for the performance prediction problem in CB WLANs (referred to as PS-013), the challenge has allowed us to glimpse the potential of ML models for addressing it.

This paper provides a compendium of ML models proposed for throughput prediction in WLANs, including popular models such as neural networks, linear regression, or random forests. In particular, we have provided an overview of the proposed models and analyzed their performance in the context of the challenge. By opening the data set used during the competition, we encourage the development of mechanisms that improve the baseline performance shown by the ML models presented in this work.

5.2 Lessons learned

From the performance evaluation done in this paper, we have drawn the following conclusions:

1. First, even if the data set was not particularly big,[6] some of the proposed ML models achieved good results. This is quite a positive result since it opens the door to ML models that can be (re)trained fast, thus becoming suitable for (near)real-time solutions.

2. Second, most of the proposed DL-based models have shown higher accuracy for the denser and more complex deployments (which more closely match the training scenarios) than for the sparser ones. While capturing complex situations is quite a positive result, the pitfalls observed in simpler deployments also suggest that out-of-the-box DL methods may fail at capturing the relationship between interference and performance of WLANs. In this regard, well-known models characterizing WLANs (e.g., SINR-based models [40]) can potentially be incorporated into the ML operation for the sake of improving accuracy, thus leading to hybrid model-based and data-driven mechanisms.

3. Third, and related to the previous point, GNNs have been shown to be particularly useful to capture the complex interactions among devices in WLANs, both in terms of interference and neighboring activity. In particular, we have realized the importance of preprocessing the data set in order to obtain accurate prediction results. Deriving information specific to the problem (i.e., signal quality, interference) has turned out to be essential for the sake of generalization.

4. Finally, we remark the importance of cost-effectively predicting the performance in WLANs, which may open the door to novel mechanisms using these predictions as heuristics for online optimization. The incorporation of these kinds of models to WLANs is expected to be enabled by ML-aware architectural solutions [41].

5.3 The role of network simulators in ML-aware communications

The availability of data for training is key for the success of ML application to future 5G/6G networks. Given the current limitations in acquiring data from real networks, simulators emerge as a practical solution to generate complementary synthetic data for training ML models. The fact is that data may be scarce because, among other reasons, measurement campaigns are costly, data from networks involves privacy concerns, or data tenants are not willing to share their data.

Through the problem statement discussed in this paper, we have contributed to showcase how synthetic data can be used to train ML models for networks. A notorious advantage is that network simulators allow characterizing complex deployments, sometimes representing unknown situations, so they help to train and validate ML models.

Beyond generating data for training, network simulators are envisioned to serve as secure platforms for testing, training, and evaluating ML models before being applied to operative networks [42]. In consequence, we foresee the adoption of simulators into future ML-aware networks as a key milestone for enhancing both reliability and trustworthiness in ML mechanisms.

[6] Youtube-8M Data set (http://research.google.com/youtube8m/) consists of 350.000 hours of video, while only six hundred different random deployments have been used in this paper for training purposes.

ACKNOWLEDGEMENT

Representative members of each participating team has been invited to co-author this paper. Nevertheless, the same credit goes to the rest of the participants of PS-013 in the ITU AI for 5G Challenge: Miguel Camelo, Natalia Gaviria, Mohammad Abid, Ayman M. Aloshan, Faisal Alomar, Khaled M. Sahari, Megha G Kulkarni, and Vishalsagar Udupi. We would also like to thank enormously everyone that made possible the ITU AI for 5G Challenge, with special mention to Vishnu Ram OV, Reinhard Scholl, and Thomas Basikolo.
This work has been partially supported by grants WINDMAL PGC2018-099959-B-I00 (MCIU/AEI/FEDER,UE), and 2017-SGR-11888.

REFERENCES

[1] ITU-T, Recommendation Y.3172 (2019). "Architectural framework for machine learning in future networks including IMT-2020".

[2] ITU-T. ITU AI/ML in 5G Challenge (2019). Available at: https://www.itu.int/en/ITU-T/AI/challenge/2020/Pages/default.aspx

[3] Wilhelmi, F., Barrachina-Muñoz, S., Bellalta, B., Cano, C., Jonsson, A., & Neu, G. (2019). "Potential and pitfalls of multi-armed bandits for decentralized spatial reuse in WLANs". Journal of Network and Computer Applications, 127, 26-42.

[4] Francesc Wilhelmi. (2020). [ITU-T AI Challenge] Input/Output of project "Improving the capacity of IEEE 802.11 WLANs through Machine Learning" [Data set]. Zenodo. http://doi.org/10.5281/zenodo.4106127

[5] Bellalta, B. (2016). IEEE 802.11 ax: "High-efficiency WLANs". IEEE Wireless Communications, 23(1), 38-46.

[6] López-Pérez, D., Garcia-Rodriguez, A., Galati-Giordano, L., Kasslin, M., & Doppler, K. (2019). "IEEE 802.11 be extremely high throughput: The next generation of Wi-Fi technology beyond 802.11 ax". IEEE Communications Magazine, 57(9), 113-119.

[7] Barrachina-Muñoz, S., Wilhelmi, F., & Bellalta, B. (2019). "Dynamic channel bonding in spatially distributed high-density WLANs". IEEE Transactions on Mobile Computing.

[8] Barrachina-Muñoz, S., Wilhelmi, F., & Bellalta, B. (2019). "To overlap or not to overlap: Enabling channel bonding in high-density WLANs". Computer Networks, 152, 40-53.

[9] Cao, R., & Zhang, H. (2019). U.S. Patent Application No. 16/435,899.

[10] Wang, W., Chen, Y., Wang, Z., Zhang, J., Wu, K., & Zhang, Q. (2016). "Wideband spectrum adaptation without coordination". IEEE Transactions on Mobile Computing, 16(1), 243-256.

[11] Huang, P., Yang, X., & Xiao, L. (2016). "Dynamic channel bonding: Enabling flexible spectrum aggregation". IEEE Transactions on Mobile Computing, 15(12), 3042-3056.

[12] Chen, Y. D., Wu, D. R., Sung, T. C., & Shih, K. P. (2018, April). "DBS: A dynamic bandwidth selection MAC protocol for channel bonding in IEEE 802.11 ac WLANs". In 2018 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1-6). IEEE.

[13] Nabil, A., Abdel-Rahman, M. J., & MacKenzie, A. B. (2017, October). "Adaptive channel bonding in wireless LANs under demand uncertainty". In 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (pp. 1-7). IEEE.

[14] Qi, H., Huang, H., Hu, Z., Wen, X., & Lu, Z. (2020). "On-demand channel bonding in heterogeneous WLANs: A multi-agent deep reinforcement learning approach". Sensors, 20(10), 2789.

[15] Pan, C., Cheng, Y., Yang, Z., & Zhang, Y. (2018, July). "Dynamic opportunistic spectrum access with channel bonding in mesh networks: A game-theoretic approach". In International Conference on Machine Learning and Intelligent Communications (pp. 381-390). Springer, Cham.

[16] Bianchi, G. (2000). "Performance analysis of the IEEE 802. 11 distributed coordination function". IEEE Journal on selected areas in communications, 18(3), 535-547.

[17] Bellalta, B., Zocca, A., Cano, C., Checco, A., Barcelo, J., & Vinel, A. (2014). "Throughput analysis in CSMA/CA networks using continuous time Markov networks: A tutorial". Wireless Networking for Moving Objects, 115-133.

[18] Baccelli, F., & Błaszczyszyn, B. (2009). "Stochastic geometry and wireless networks (Vol. 1)". Now Publishers Inc.

[19] Feng, H., Shu, Y., Wang, S., & Ma, M. (2006, June). "SVM-based models for predicting WLAN traffic". In 2006 IEEE International Conference on Communications (Vol. 2, pp. 597-602). IEEE.

[20] Abbas, M., Elhamshary, M., Rizk, H., Torki, M., & Youssef, M. (2019, March). "WiDeep: WiFi-based accurate and robust indoor localization system using deep learning". In 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom (pp. 1-10). IEEE.

[21] Davaslioglu, K., Soltani, S., Erpek, T., & Sagduyu, Y. (2019). "DeepWiFi: Cognitive WiFi with deep learning". IEEE Transactions on Mobile Computing.

[22] Khan, M. A., Hamila, R., Al-Emadi, N. A., Kiranyaz, S., & Gabbouj, M. (2020). "Real-time throughput prediction for cognitive Wi-Fi networks". Journal of Network and Computer Applications, 150, 102499.

[23] Barrachina-Muñoz, S., Wilhelmi, F., Selinis, I., & Bellalta, B. (2019, April). "Komondor: a wireless network simulator for next-generation high-density WLANs". In 2019 Wireless Days (WD) (pp. 1-8). IEEE.

[24] Wilhelmi, F., Barrachina-Muñoz, S., Cano, C., Selinis, I., & Bellalta, B. (2021). "Spatial reuse in IEEE 802.11 ax WLANs". Computer Communications.

[25] ITU-T, ML5G-I-237 (2020). "A compilation of problem statements and resources for ITU Global Challenge on AI/ML in 5G networks". Available online: https://www.itu.int/en/ITU-T/AI/challenge/2020/Documents/ML5G-I-237-R5_v10.docx

[26] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The graph neural network model," IEEE Transactions on Neural Networks, pp. 61–80, 2008.

[27] P. Battaglia, J. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro and R. Faulkner, "Relational inductive biases, deep learning, and graph networks". arXiv preprint arXiv:1806.01261, 2018.

[28] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[29] P. Soto, D. Góez, M. Camelo, N. Gaviria and K. Mets, "ITU AI/ML in 5G Challenge 2020: Team ATARI (Github repository)", 12 Nov 2020. Online: https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-013-ATARI. [Accessed 20 Feb 2021].

[30] Murtagh, F. (1991). "Multilayer perceptrons for classification and regression". Neurocomputing, 2(5-6), 183-197.

[31] R. Vallés (2020), "MLP Throughput prediction (Github repository)". Online: https://github.com/VPRamon/MLP-Throughput-prediction. [Accessed 20 Feb 2021].

[32] Chen, T., & Guestrin, C. (2016, August). "Xgboost: A scalable tree boosting system". In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[33] Yeo, I-K, and R Johnson. 2000. "A New Family of Power Transformations to Improve Normality or Symmetry." Biometrika 87 (4): 954–59.

[34] Kuhn M, Wickham H (2020). "Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles". Available online: https: //www.tidymodels.org.

[35] M Alfaifi, A Algunayah, A Aloshan, M Abid & K Sahari (2020). "ITU 2020: Problem 13 - STC Team 2". Github repository. Available online: https://github.com/ITU-AI-ML-in-5G-Challenge/p13_stc_team2. [Accessed on 21 February 2021]

[36] Jorge Martin Pérez & Luigi Girletti (2020). "ITU-ML5G-PS-013", Github repository. Available online: https://github.com/MartinPJorge/ITU-ML5G-PS-013. [Accessed on 21 February 2021]

[37] Rajasekar Mohan, K Venkat Ramnan, Megha G Kulkarni, and Vishalsagar Udupi (2020), "Net Intels solution for the ITU-T AI Challenge (Github repository)". Online: https://github.com/venkatramnank/UPF_ITU_T_5G_ML_AI_Challenge_NetIntels. [Accessed 20 Feb 2021].

[38] Brownlee, J. (2016). "Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras". Machine Learning Mastery.

[39] Pedregosa, F., et al. (2011). "Scikit-learn: Machine learning in Python". The Journal of Machine Learning Research, 12, 2825-2830.

[40] Yang, X., & Vaidya, N. (2005, March). "On physical carrier sensing in wireless ad hoc networks". In Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. (Vol. 4, pp. 2525-2535). IEEE.

[41] Wilhelmi, F., Barrachina-Muñoz, S., Bellalta, B., Cano, C., Jonsson, A., & Ram, V. (2020). "A flexible machine-learning-aware architecture for future WLANs". IEEE Communications Magazine, 58(3), 25-31.

[42] Wilhelmi, F., Carrascosa, M., Cano, C., Jonsson, A., Ram, V., & Bellalta, B. (2020). "Usage of Network Simulators in Machine-Learning-Assisted 5G/6G Networks". IEEE Wireless Communications Magazine, 2021.

AUTHORS

Francesc Wilhelmi holds a Ph.D. in information and communication technologies (2020) from Universitat Pompeu Fabra (UPF). He is currently a postdoctoral researcher in the Mobile Networks department at Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) and a teaching assistant at UPF and at Universitat Oberta de Catalunya (UOC).

David Góez is a telecommunications engineer from the ITM Metropolitan Technological Institute, a Master in Automation and Industrial Control from the ITM, a PhD student in electronic and computer engineering at the University of Antioquia. His research focuses on radiocommunication systems whose signal processing blocks can be built with machine learning.

Paola Soto is currently pursuing her Ph.D. degree at University of Antwerp - imec, Belgium. Her doctoral research investigates machine learning techniques applied to network management.
She received a B.Sc. degree in Electronics and an M.Sc. degree in telecommunications engineering from the University of Antioquia, Medellín, Colombia, in 2014 and 2018, respectively. Her main research interests include artificial intelligence, machine learning, network management, and resource allocation algorithms.

Ramon Vallés is currently pursuing a B.Sc. in computer science at Universitat Pompeu Fabra (UPF).
His research interests are artificial intelligence and machine learning, networks and network security.

Mohammad Alfaifi holds an M.Sc. degree in management studies from the University of Waikato (New Zealand). He is currently working as Data Mining Supervisor at Saudi Telecom (STC).

Abdulrahman Algunayah holds a B.Sc. degree in electrical engineering from King Saud University (KSU).
He is an infrastructure design engineer at Saudi Telecom (STC).

Jorge Martín-Pérez is a telematics engineering Ph.D. student at Universidad Carlos III de Madrid (UC3M). He obtained a B.Sc. in mathematics and computer science, both in 2016 from Universidad Autónoma de Madrid. In 2017, he obtained an M.Sc. in telematics engineering from UC3M. His research focuses on optimal resource allocation in 5G networks, and since 2016 he participates in EU-funded research projects.

Luigi Girletti is a telematics engineering Ph.D. student at Universidad Carlos III de Madrid (UC3M). He obtained both a B.Sc. and an M.Sc in computer science engineering in 2014 and 2018 from Universitá degli Studi Federico II di Napoli. His research focuses on SDN/NFV, artificial intelligence applied to next-generation networks, and 5G-based EU-funded research projects.

Rajasekar Mohan, alumnus of Indian Institute of Technology, Madras, is pursuing his research in the domain of wireless communication at PES University. His research interests include embedded systems, IoT, robotics and application of ML in communication. He has served in the Indian Air Force for over 23 years in various technology roles in the field of communication. Currently, he is an associate professor in the department of ECE at PES University, Bangalore.

K Venkat Ramnan is an undergraduate student in the Electronics and Communication Department at PES University, India. He is a beginner in machine learning, deep learning and image processing. He also works with the Internet of things, networking and Linux. He aims to apply machine learning and deep learning in various fields including wireless communications, healthcare and various other domains of electronics and communications.

Boris Bellalta is an associate professor in the Department of Information and Communication Technologies (DTIC) at Universitat Pompeu Fabra (UPF). He is the head of the Wireless Networking research group at DTIC/UPF. His research interests are in the area of wireless networks, with emphasis on the design and performance evaluation of new architectures and protocols.

AI-BASED NETWORK TOPOLOGY OPTIMIZATION SYSTEM

Han Zengfu, Kong Jiankun, Wang Zhiguo, Zhang Yiwei, Liu Ke, Pan Liang, Li Sicong, Wu Desheng

China Mobile Shandong Branch, 20569, Jingshi road, Jinan, Shandong

NOTE: Corresponding author: Han Zengfu, hanzengfu@sd.chinamobile.com

Abstract *– Existing network topology planning does not fully consider the increasing network traffic and problem of uneven link capacity utilization, resulting in lower resource utilization and unnecessary investments in network construction. The AI-based network topology optimization system introduced in this paper builds a Long Short-Term Memory (LSTM) model for time series traffic forecasting, which uses NetworkX, a Python library, for graph analysis, dynamically optimizes the network topology by edge deletion or addition based on traffic over nodes, and ensures network load balancing when node traffic increases, mainly introducing the LSTM forecasting model building process, parameter optimization strategy, and network topology optimization in some detail. As it effectively enhances resource utilization, this system is vital to the optimization of complex network topology. The end of this paper looks forward to the future development of artificial intelligence, and suggests the possibility of how to cooperate with operator networks and how to establish cross-border ecological development.*

Keywords –Artificial intelligence, capacity utilization, communication network, traffic forecast, network topology

1. INTRODUCTION

With the development of 5G technology, operators need to rebuild or expand their transport networks, as their existing network topology planning does not fully consider the increasing network traffic and problem of uneven link capacity utilization [1]. The utilization of more than 40% of the transmission links of the existing networks is low [2,3], which increases operators' network construction costs. Therefore, how to optimize the network structure to make it adapt to future network changes and how to improve resource utilization to save construction costs have greatly challenged today's operators [7].

For complex networks, topology analysis requires exponential computing [4,5], making manual network topology optimization very difficult. With the development of Artificial Intelligence (AI) technology, Machine Learning (ML) algorithms offer a helping hand to resolve complex issues [6,7]. A new ecosystem which needs to be established to advance such technical developments as applying AI and big data technologies to networks is new to the industry, and the "AI-ML in 5G Challenge" organized by ITU paves the way for such study. This paper introduces an AI-based network topology optimization system that makes in-depth analysis of network topology and proposes a solution to maintain high resource utilization when network traffic keeps growing [2]. It is especially crucial for the optimization of complex network topology.

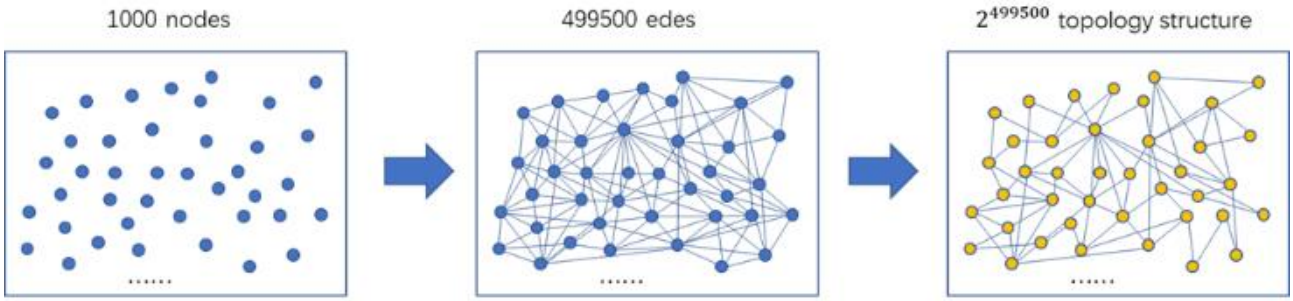


Fig. 1 – Complicated network topology analysis

2. LSTM MODEL FOR TRAFFIC FORECASTING

Network traffic forecasting is made based on network elements and affected by various factors such as the location of network elements, weather, and traffic on different base stations. Forecasting is mainly implemented via Gompertz models or time series forecasting models like LSTM and Prophet models [8,9,10]. The traffic forecast made based on traditional Extended Gompertz Models (EGMs) cannot reflect traffic differences between working days and holidays. However, the traffic predicted based on time series forecasting models with deep learning is more accurate, as history data can be learned and used during the traffic forecast [11].

2.1 Traffic forecasting model with deep learning

The short-term traffic forecast predicts the data of the next few days. As daily traffic features strong periodicity, traffic curves of the days in weeks with similar attributes are almost the same. In view of that, this paper proposes an LSTM model [10,11], a recurrent neural network architecture that gives full play to the correlation of traffic at different times, for traffic forecasting. The model is trained with time-stamped traffic (for example, traffic of working days or holidays), and outputs more accurate predicted data through iteration technologies [12].

2.2 Establishment of the LSTM model

In the study, the LSTM model was built with TensorFlow2.0 and traffic per hour was taken as a sample. The traffic samples of the last 20 days were processed first, and the traffic over each network element was listed by time series after processing [13,14]. The following four steps describe how the LSTM model was built and traffic forecasting was conducted.

1) Make data sets: The traffic samples of the first 15 days were used for training and testing, and those of the remaining 5 days were used for result verification. The training data set included 75% traffic samples of the first 15 days, and the testing data set contained the other 25%. The initial default time sliding window was 24.

2) Establish the LSTM model: Add two LSTM layers to the LSTM model for traffic forecasting.

3) Train the model: Input training data and test data into the LSTM model, and set epochs and batch_size to 50 and 32 respectively.

4) Conduct traffic forecasting: Forecast the traffic of the next hour through iteration technologies based on the time sliding window. Enhance the forecasting efficiency via multiprocessing technologies and concurrent processes.

2.3 Optimization of the LSTM model

At the beginning of the study, the traffic forecasted with the default parameters of the model was quite different from the data (traffic of the last five days) which remained for result verification. After repeated comparison and analysis, the most proper epochs, batch_size, and slide_window were determined. The following shows the way that we used in the study to determine those parameters.

First, set the batch_size and slide_window to fixed values, and the epochs to 50, 100, 200 and 400 respectively. Then, evaluate the impact of each epoch's value on the errors in traffic forecasting in terms of the Root Mean Square Error (RMSE) and running time. The results indicate that "epochs=200" is the optimal choice.

After the model was trained for 50, 100, 200, and 400 times respectively, the RMSE generated accordingly, predicted traffic volume, and the time that training costs were recorded in the following table.

Table 1 – Optimization of the model's epochs

Times	RMSE	Predicted Traffic Volume (GB)	Running Time(s)
50	16.246	251.3	55
100	3.714	279.9	85
200	1.9	292.4	148
400	2.3	293	305

As Table 1 shows, when the model was trained for 200 times, it generated the smallest RMSE and cost the shortest time. However, when the training was conducted for 50, 100, and 400 times, both the RMSEs that were generated accordingly, and the running time of each training failed to meet the requirements. Therefore, the model was finally trained for 200 times to ensure both high efficiency and accuracy.

When epochs were set to 200 and slide_window to a fixed value, the result of repeated comparison of the errors caused by configuring different batch_size showed that batch_size=64 is the best choice.

When epochs=200 and batch_size=64, "slide_window=48" causes the minimum error in traffic forecast.

Fig. 2 shows the errors in traffic forecast when the model is configured with different parameter values.

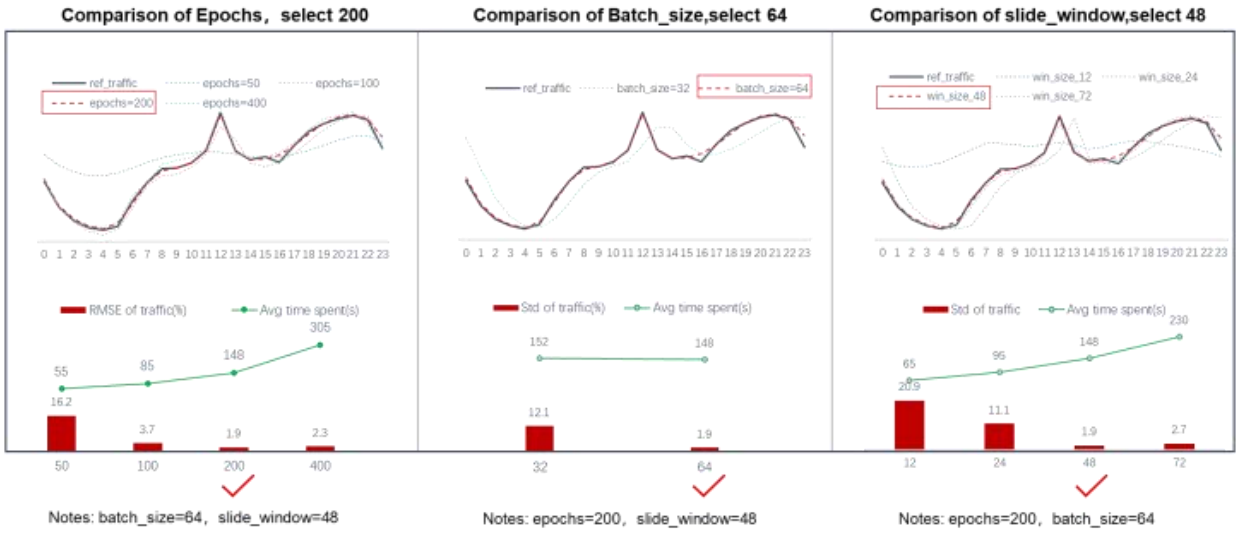


Fig. 2 – Model set with different parameter value

In view of accuracy of the forecast and running efficiency, the parameter values of the traffic forecasting model were selected through multiple rounds of testing. The final values used in the system ensure that the average error in traffic forecast is under 3%, as shown in Fig. 3 below.

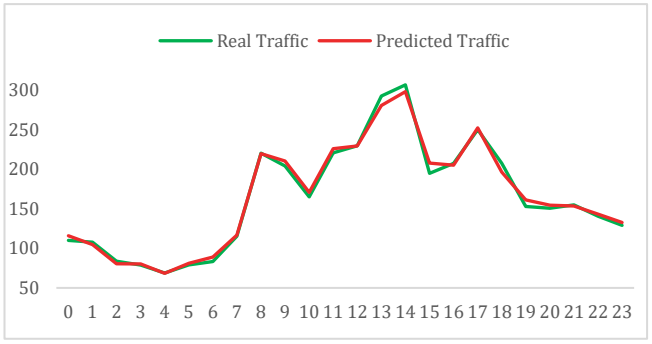


Fig. 3 – Comparison between predicted data and the data for result verification

After optimization, we input the same sample into the new LSTM model (which was named Our-LSTM) and other five well-known models respectively, namely ARIMA, LightGBM, Prophet, LSTM, and DeepAR, and made some comparisons. For specific information, refer to Table 2.

Table 2 - Comparisons of traffic forecasting models

Model	Absolute Accuracy (%)	Relative Accuracy
ARIMA	18.36	2.6721
LightGBM	20.31	1.8742
Our-LSTM	3.01	0.6552
Prophet	8.88	2.3516
LSTM	15.02	1.7471
DeepAR	16.3	1.6913

As Table 2 shows, both the absolute and relative accuracy achieved through Our-LSTM are better than those achieved based on the initial LSTM, proving that the hyper-parameters of Our-LSTM after optimization are more suitable for the network. When compared with other time series forecasting models listed in the above table, Our-LSTM with the highest absolute accuracy and relative accuracy is superior to them in short-term time series forecasting.

3. NETWORK TOPOLOGY ANALYSIS FOR OPTIMIZATION

Topology optimization is quite complicated. Existing network topology planning can hardly satisfy future traffic increase or solve the problem of uneven load balancing, as it is made based on the current network load. Our research is to resolve this problem through dynamic network topology optimization. Specifically, the topology optimization system introduced in this paper enables network load balancing and makes the topology satisfy future traffic growth.

NetworkX, a Python library for studying graphs and networks, is an effective tool for analyzing network topology, building network models, designing new network algorithms, and plotting network graphs. In this system, NetworkX is used to plot the network graph [15].

3.1 Network topology reshaping

3.1.1 Define nodes and links

As per different functions, significance, and capacity, the nodes in the network topology to be optimized can be classified into three types, namely, nodes G, H, J [16,17,18]. There are also three types of links, namely, main links, sub-links, and hanging links.

Link: A link in the network topology can be made of one main link, zero or multiple sub-links, and zero or multiple hanging links. The total number of nodes on each link does not exceed 30.

Main link: Each main link has the following features: 1) Its end nodes are node G/H. 2) Its intermediate nodes are node H or J. 3) The capacity of each intermediate node is the same. 4) The total number of nodes on one main link equals to or is less than 15.

Sub-link: Each sub-link complies with the following requirements: 1) Its end nodes are node G/H/J on main links (When the types of the start node and end node are different, they can only be node G/H), and intermediate nodes should be node J. 2) The capacity of a node J is smaller than or equal to that of either end node of the sub-link.

Hanging link: The link connects with a node G, or H, or J on one main link or sub-link through one edge only. The capacity of each node on a hanging link is smaller than or equal to that of the node where it connects, as shown in Fig. 4 below.

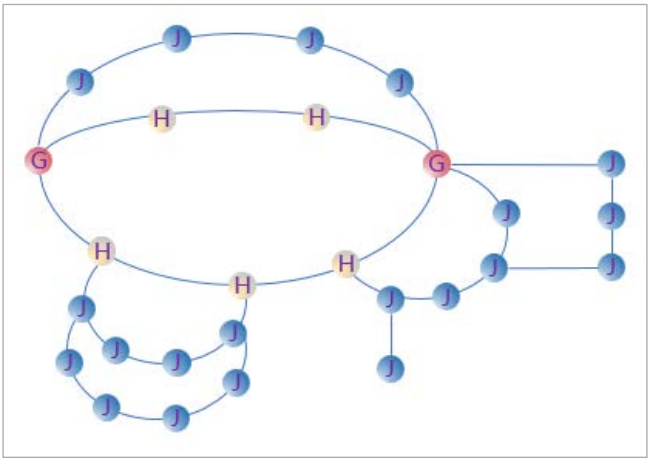


Fig. 4 – Structure of network topology

3.1.2 Introduction of neighbor links and neighbor nodes for faster topology analysis

Neighbor nodes and neighbor links in the network graph created with NetworkX can be defined [19]. As for neighbor nodes, one node on a specified link is adjacent not only to the appropriate node on the same link, but also to the nodes within a certain distance. Neighbor links refer to the links where two neighbor nodes (on different links) locate. The network topology in full status indicates that all neighbor nodes in this topology are connected. Such topology includes both existing links and all potential neighbor links. Network topology in optimized status refers to the network graph that we achieve by removing appropriate edges. Topology in full status and in optimized status can be mutually transformed.

Neighbor nodes and neighbor links become the only two factors that one needs to figure out when any node or link is to be analyzed, making iteration more simplified and topology analysis more efficient, as shown in Fig. 5 below.

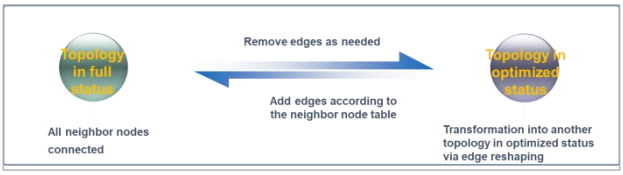


Fig. 5 – Full and optimized status

3.2 Innovative "Node-Removing Method" for highly efficient topology recovery

The network topology solution proposed in this paper is implemented based on edge reshaping. Specifically, links in the network topology need to be sorted out and saved in a link library.

First, find the main links whose end nodes are G or H via the DFS algorithm [20,21], and save all such main links in the link library. For cross-connected main links, select the one that is comparatively longer than the others in accordance with constraint rules on cross-connected links.

Regarding the sub-links and hanging links, the innovative "Node-Removing Method" is applied to sort them out when all main links have been identified. Specifically speaking, after main links in the network topology are removed, the related sub-links and hanging links will be disconnected accordingly. In this way, all the links in this network topology can be sorted out by category, and all the links of the topology will be recorded in the library by category.

1) Select one main link from the link library, and then remove it by using NetworkX.
2) Execute the connected_components() method to identify all connected sub-graphs [15].
3) Review all these sub-graphs and remove the sub-graphs that are not related to the current main link in accordance with the existing node connections.
4) Associate the sub-links and hanging links in the sub-graphs that have been removed in step 3 with the correct main link, and update the collection of neighbors' links in the link library.
5) Continue to remove other main links in the same way until all the sub-links and hanging links in the graph are associated with appropriate main links, as shown in Fig. 6 below.

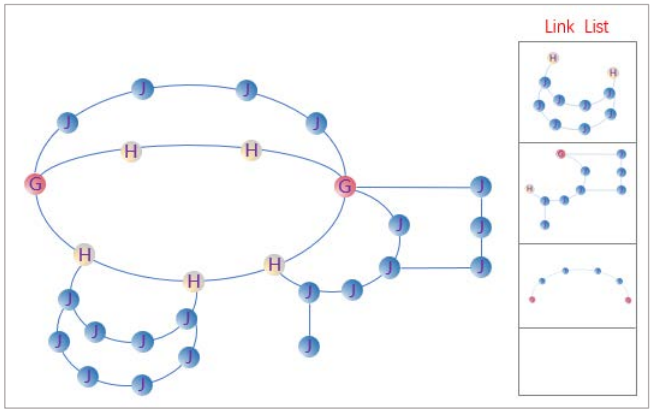


Fig. 6 – Node removing method

3.3 Evaluation of network topology

The following formula describes how the effect of network topology optimization is evaluated based on the network service rules, rules for assessing node levels [22,23], and the requirements for link load balancing.

$$\max \text{object\_ratio}_{avg} - (E_{avg}+E_{min}+E_{max}) - \alpha \cdot \text{sub\_ratio} - \beta \cdot \text{hang\_ratio} \quad (1)$$

Explain:

$\text{object\_ratio}_{avg}$: Daily average of the ratio of links whose bandwidth utilization is optimized to the target range per hour

E_{avg}: Daily average of the E-value per hour that indicates network load balancing

E_{min}: Minimum E-value per hour of each day

E_{max}: Maximum E-value per hour of each day

sub_ratio: Proportion of nodes on sub-links to those on the entire link (daily average of values per hour)

hang_ratio: Proportion of nodes on hanging links to those on the entire links (daily average of values per hour)

α,β: Constant coefficients that are set to 0.02 and 0.05 respectively.

Edge addition: Distance between two nodes that are to be connected by adding an edge for topology optimization should be within 500 meters.

Number of nodes on each link: Maximally 30 nodes on each link.

Target range of link bandwidth utilization: $[\frac{3}{5}\,\mu\,,\ \ \frac{7}{5}\,\mu]$

Formula used to calculate E-value:

$$f_p = \frac{\sum_i^{nodes\ number} f_{xi}}{A} \quad (2)$$

"E" is the variance of the link bandwidth utilization (fp) of all links in the network topology.

"μ" is the average of bandwidth utilization of all the links in the network topology. In the formula, fx is the traffic on the nodes (excluding the start and end nodes) on the link, and A stands for the maximum node capacity that one node (excluding the start and end nodes) on the link can possess.

3.4 Build and complete network topology optimization and find the optimal solution through iteration

Network topology optimization for load balancing can be implemented in three different ways [24,25,26,27], including link combination, partial link optimization, and the optimization by node transfer.

Identify links with heavy or low load based on their link utilization, check the link library to search for their neighbor links that carry low load or whose link utilization is within the target range, confirm whether any link combination can be conducted in accordance with the rules for link optimization, and combine the links that comply with the rules.

In the case where links cannot be optimized in the above way, partial link optimization can be implemented. 1) Identify the links adjacent to the sub-links or hanging links of the links with heavy load; 2) Opt for one appropriate neighbor link and connect those sub-links or hanging links with the neighbor link in accordance with the related rules.

The optimization by node transfer is applicable to all the links (including those whose link bandwidth utilization is within the target range). Specifically, the links are optimized through the transfer of some nodes with heavy load on these links to other links. It reduces the variance in the utilization of all links.

During network topology optimization, the iteration of network topology is carried out every hour, which is 24 times a day.

Regarding the topology optimization system, the network topology iteration carries out every hour after the first topology recovery. Therefore, each day, 25 scores on link 'status calculated' based on the evaluation formula can be achieved. During the topology iteration, the previous day's network topology with the highest score is used to start the next round of network topology optimization. Finally, the result of the optimization showed that the proportion of links with balanced load increased by 86%, as shown in Fig. 7 below.

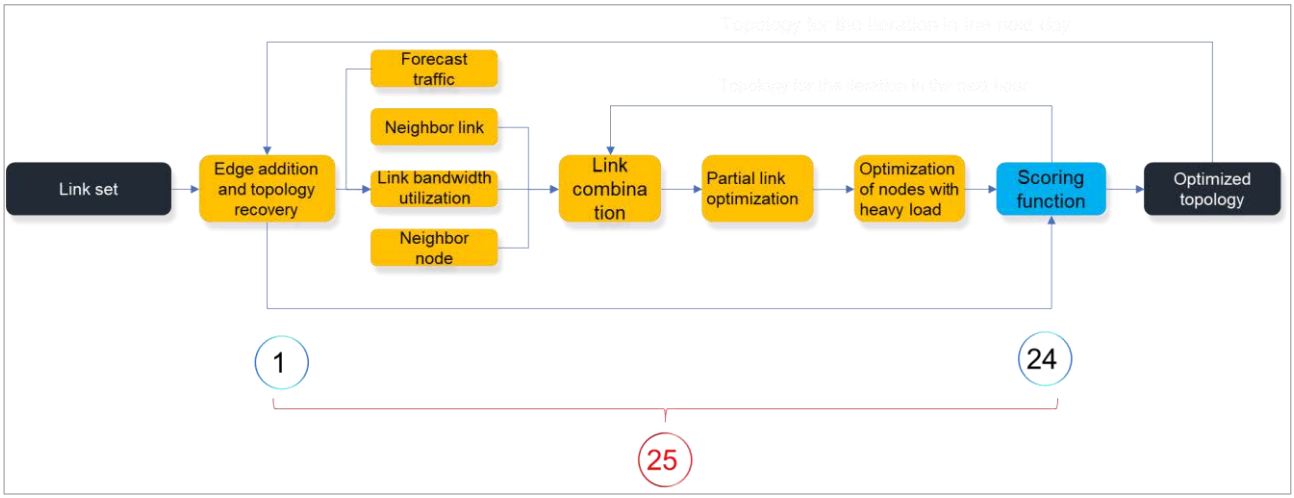


Fig. 7 – Network topology optimization process

3.5 Unique network topology restructuring for better topological structure

Network topology restructuring is to optimize the links that carry heavy or low load for a long time when basic network topology optimization is completed [28,29].

First, identify links with a heavy or low load for a long time. Second, split the link with heavy load into two links through either of the following two ways. One is to connect the link with two neighbor nodes (node G or H) respectively by adding two new edges. The other is to connect it with another main link that carries low load. By doing so, the iteration of the new topology carried out afterwards based on the new structure improves the utilization of all links [30].

The network topology restructuring aims to optimize network topology in a more complete way. In other words, it changes the status of the links that carry unbalanced load for a long time. In the study, we extended the distance between two neighbor nodes to 1000 meters and witnessed a 169% increase in the proportion of the links with a balanced load through the network topology restructuring [31,32], as shown in Fig. 8 below.

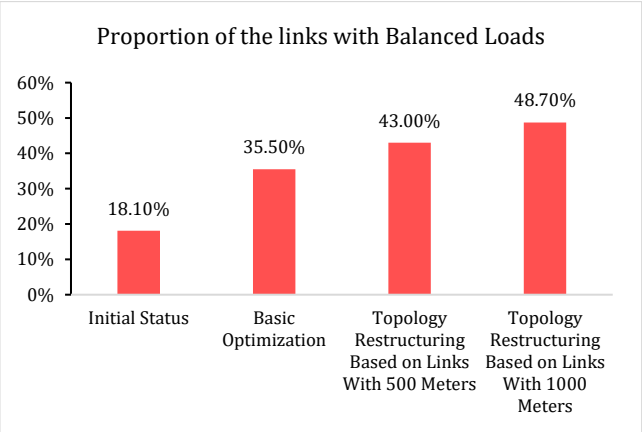


Fig. 8 – Proportion of links with balanced load in different optimization stages

The above-mentioned solution introduces a complete network topology optimization system for effective resolution of network topology problems, as shown in Fig. 9 below.

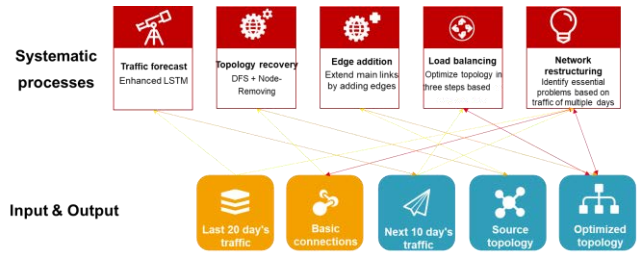


Fig. 9 – A complete network topology optimization system

4. CONCLUSION

Traffic forecasting and network load balancing are always under the spotlight of operators. During the study, all tests prove that the forecast accuracy and efficiency of the optimized LSTM are higher than those of ARIMA, LightGBM, Prophet, and DeepAR. Therefore, it can provide better support for operators' resource investment. The network topology optimization model proposed in this paper is optimized based on the actual size of the network that we worked on in the study. In view of different locations, periods of time, network sizes, and network characteristics, the model needs to be optimized in accordance with different topology reshaping rules tailored based on the nodes and routing of the network to be optimized. This model is vital to network topology optimization, as it can enhance the resource utilization by over 30%.

In this paper, a brand-new AI-based network topology optimization system is proposed. By analyzing future network traffic development trends via a traffic forecast model and refining network structures through a network topology optimization model, this system enhances the utilization of network resources and reduces operators' investments. The advent of the 5G era brings the world booming network traffic and more complicated network structures, and AI-based network topology optimization is superior to other methods for its outstanding capability and practicability in dealing with complex networks. It ensures sustainable network development and guarantees the return on investment ratio for operators. The AI-based network topology optimization system introduced in this paper is proposed based on ITU AI standards, and features:

1) Creativity: It addresses difficulties and promotes the development of intelligent networks by applying cutting-edge AI technologies to operator networks.

2) Enhanced traffic forecast algorithm: It enables more accurate traffic forecasting.

3) A complete network topology analysis and optimization structure: The concepts of neighbor, node removing method, and three-step network topology optimization that are introduced for the first to the industry effectively accelerate the network topology analysis. Moreover, the network restructuring fixes the defects of existing network topologies, paving the way for future network topology optimization.

The challenge organized by ITU has offered a great opportunity for building a cross-field ecosystem, and operators and ITU should continue to make joint efforts (e.g. encourage crowd-funding AI algorithms) to resolve common problems. For example, building a middleware platform to open data and solve problems together, organizing more competitions and building the ecosystem for developers for closer collaboration.

REFERENCES

[1] Lampiris E., Zhang J., Simeone O., et al. Fundamental Limits of Wireless Caching under Uneven-Capacity Channels[J]. 2019.

[2] He Z., Jian H. Application of Multilink Aggregation and Load Balancing in Wireless Real-Time Video Transmission System[C]// 2018 International Conference on Sensor Networks and Signal Processing (SNSP). IEEE Computer Society, 2018.

[3] Wang Y., Gong B., Gong M. Research on the Trusted Energy-Saving Transmission of Data Center Network. China Communications, 2016.

[4] Kamiyama N. Network Topology Design Using Data Envelopment Analysis[C]// IEEE Globecom-ieee Global Telecommunications Conference. IEEE, 2017.

[5] Liu Z., Zou Z. Analysis of network topology and deployment mode of 5G wireless access network[J]. Computer Communications, 2020, 160.

[6] Singh J P. THE ROLE OF AI, ML, AND IOT IN DIGITAL TRANSFORMATION IN 2019[J]. PC Quest, 2019, 32(1):18-19.

[7] Liu Z., Yubo M U., Zhang Y. Overview of network artifical intelligence requirements and applications. Telecommunications Network Technology, 2018.

[8] Weytjens H., Lohmann E., Kleinsteuber M. Cash flow prediction: MLP and LSTM compared to ARIMA and Prophet[J]. Electronic Commerce Research, 2019(1).

[9] Trappey C V., Wu H Y. An evaluation of the time-varying extended logistic, simple logistic, and Gompertz models for forecasting short product lifecycles[J]. Advanced Engineering Informatics, 2008, 22(4):421-430.

[10] Greff K., Srivastava R K. , J Koutník, et al. LSTM: A Search Space Odyssey[J]. IEEE Transactions on Neural Networks and Learning Systems","pubMedId":"27411231, 2017.

[11] Do We Really Need Deep Learning Models for Time Series Forecasting?, 2021.

[12] Gers F. A., Eck D., Schmidhuber J. Applying LSTM to Time Series Predictable through Time-Window Approaches[J]. Springer, Berlin, Heidelberg, 2001.

[13] Moustapha A. I., Selmic R. R . Wireless Sensor Network Modeling Using Modified Recurrent Neural Networks: Application to Fault Detection[J]. IEEE Transactions on Instrumentation & Measurement, 2008, 57(5):981-988.

[14] Sang C., MD Pierro. Improving trading technical analysis with TensorFlow Long Short-Term Memory (LSTM) Neural Network - ScienceDirect[J]. The Journal of Finance and Data Science, 2019, 5(1):1-11.

[15] Hagberg A., Schult D., Swart P. NetworkX Tutorial. 2011.

[16] Chuan W. W., Zhang B. M. A GRAPHIC DATABASE BASED NETWORK TOPOLOGY AND ITS APPLICATION[J]. Power System Technology, 2002.

[17] Chen L. W., Martin N., Cabrera C., et al. Communication network topology determination: US, US20110188409 A1[P]. 2011.

[18] Wang H., Chen Y. Network topology description and visualization[J]. IEEE, 2010.

[19] O'Brien P. S. Optimizing hand-off neighbor lists for improved system performance[J]. 2004.

[20] Sánchez-Torrubia M. G., Torres-Blanc C., Cubillo S. Design Of A Fuzzy Inference System For Automatic DFS & BFS Algorithm Learning Assessment [M]. 2015.

[21] Vijay I., Garg K. Algorithm DFS(S): begin. 2003.

[22] Yang C., Ma J., Yao Z., et al. Method for routing mobile node in wireless mesh network and a communication system thereof [J]. 2012.

[23] Wang W., Yang W., Yang C., et al. Method For Routing Mobile Node In Wireless Mesh Network And A Communication System Thereof [J]. 2008.

[24] Jinsong Gui, Kai Zhou. Flexible Adjustments Between Energy and Capacity for Topology Control in Heterogeneous Wireless Multi-hop Networks. Central South University，2016.

[25] Rui Ma, Xianghui Cao, Shuai Zhang, Lu Liu, Yu Cheng and Changyin Sun. A DBN-based Independent Set Learning Algorithm for Capacity Optimization in Wireless Networks. Southeast University，2018.

[26] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee and Nick McKeown. ElasticTree: Saving Energy in Data Center Networks. Stanford University. 2010.

[27] Bryan Perozzi, Rami Al-Rfou, Steven Skiena. DeepWalk: Online Learning of Social Representations. Stony Brook University Department of Computer Science. 2014.

[28] Danyang Zhuo, Monia Ghobadi, Ratul Mahajan, Klaus-Tycho Förster, Arvind Krishnamurthy and Thomas Anderson. Understanding and Mitigating Packet Corruption in Data Center Networks. niversity of Washington. 2017

[29] Krzysztof Rusek, José Suárez-Varela, Albert Mestres,Pere Barlet-Ros and Albert Cabellos-Aparicio. Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN.

[30] Zhang Q., Chen G., Liang Z., et al. Topology constructing and restructuring mechanisms for Bluetooth radio networks[C]// 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW). IEEE, 2016.

[31] Chiang M., Ou C., Wang J., et al. Method and apparatus for configuring a network topology with alternative communication paths: US, US7693072 B2[P]. 2010.

[32] Missaoui, Nègre, Anggraini, et al. Social network restructuring after a node removal[J]. International Journal of Web Engineering & Technology, 2013, 8(1).

AUTHORS

Han **Zengfu** received his bachelor's degree in applied physics from Beijing Jiaotong University in 2000, and his master's degree in electromagnetic field and microwave technology from the same university 5 years later. His research interests focus on machine learning theory and algorithms. In 2020, Weeny Wit, the team led by Mr. Han, won the Gold Champion (1st prize) in ITU AI/ML 5G Challenge.

Kong Jiankun received his bachelor's degree in electronic engineering from Shandong University in 1993, and his master's degree in Radio Electronics from Peking University in 1996.

Wang Zhiguo received his bachelor's degree in telecommunications engineering from Chongqing University of Posts and Communications in 2002, and his master's degree in electromagnetic fields and microwave technology from the same university in 2005. His research interests focus on mobile communications, electromagnetic field and microwave technology, and 5G network optimization, etc.

Zhang Yiwei received her bachelor's degree in electrical engineering and automation from the Northeast Forestry University in China in 2017 and her master's degree in telecommunications engineering from the University of Wollongong (UOW) in Australia in 2019. After graduation, she worked in the field of wireless communications, with her research interests focusing on network optimization. At the end of 2020, Weeny Wit, the team where Ms. Zhang was a key member, won the Gold Champion (1st prize) in ITU AI/ML 5G Challenge.

Liu Ke received his bachelor's degree in telecommunications engineering from Shandong University in 1999. His research interests include mobile communications theory and 5G network optimization, etc.

Pan Liang received his master's degree in signal and information processing engineering from Shandong University in 1999.

Li Sicong received his bachelor's degree in NanJing University of Posts and Communications of China in 2014. Since then he has been working in China Mobile Shandong Branch, responsible for LTE and 5G wireless optimization. His research interests include LTE/5G wireless networks with a focus on principles and signaling and protocols. He has been awarded by China Mobile for outstanding representation in 2017 and 2020's competition. Especially in 2020 where he won the 1st prize in the group competition, 2nd prize in the individual competition.

Wu Desheng graduated from Shandong University in 2009, and now works in the Network Department of China Mobile Shandong Branch, responsible for the maintenance and optimization of 4/5G networks.

APPLYING MACHINE LEARNING IN NETWORK TOPOLOGY OPTIMIZATION

Zhouwei Gang, Qianyin Rao , LinGuo, LinXi, Zezhong Feng, QianDeng
GuanShanXi Road, Guanshan Lake District, GuiYang GuiZhou, China

NOTE: Corresponding author: Zhouwei Gang (13595101082@139.com)

Abstract *– Nowadays, telecommunications have become an indispensable part of our life, 5G technology brings better network speeds, helps the AR and VR industry, and connects everything. It will deeply change our society. Transmission is the vessel of telecommunications. While the vessel is not so healthy, some of them are overloaded, meanwhile, others still have lots of capacity. It not only affects the customer experience, but also affects the development of communication services because of a resources problem. A transmission network is composed of transmission nodes and links. So that the possible topology numbers equal to node number multiplied by number of links means it is impossible for humans to optimize. We use Al instead of humans for topology optimization. The AI optimization solution uses an ITU Machine Learning (ML) standard, Breadth-First Search (BFS) greedy algorithm and other mainstream algorithms to solve the problem. It saves a lot of money and human resources, and also hugely improves traffic absorption capacity. The author comes from the team named"No Boundaries". The team attend ITU AI/ML in 5G Challenge and won the Gold champions (1st place).*

Keywords – 5G, artificial intelligence (Al), data handling, intelligence level,machine learning (ML)

1. BACKGROUND

Nowadays, telecommunications have become an indispensable part of people's lives. We connect to the Internet through mobile phones for social networking (Facebook, Weibo, etc.), entertainment (YouTube, mobile games, etc.), e-shopping (Amazon, eBay, etc.) and work (remote office OA, etc.), and the upcoming 5G network has expanded from people-to-people communication to people-to-things, and things-to-things communication. Therefore, the future network will be more deeply embedded in our society and life. Assuming that a tele-communication system is a human body, transmission is as important as a vessel system which sends our data (also known as traffic) to the place it is needed. To construct a highly efficient network, each node should be used efficiently to form the transmission network.

However, just as a large number of diseases are related to a blood vessel， the situation of the transmission network is not optimistic in the actual communication network.

Due to the characteristics of mobile communications, increasingly complex mobile Internet services, and increasing traffic, the utilization rate of the transmission network is generally unbalanced. Congestion is caused by the overload of a small number of key nodes, and about half of the nodes are under low load. Take the China Mobile Kaili PTN network as an example. Among the 913 links in the entire network, high-load links account for 9% while low-load links reach 53%. Operations staff need to optimize the network topology.

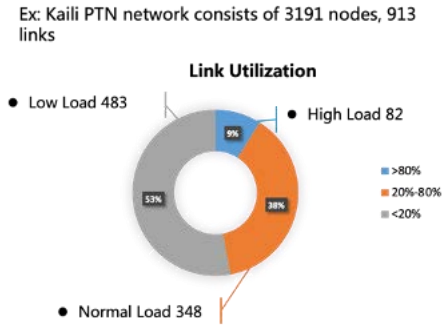


Fig. 1 – Link utilization

A transmission network is composed of transmission nodes and links. So that the possible topology numbers equal the node number multiplied by the number of links, taking Kaili as an example, the possibility of all topologies exceeds 2.9 million, which is impossible for humans to optimize.

So, at present, we can only adopt two methods, one is to adjust only one link locally. We optimize some of the chains with ultra-high utilization rates by experience or artificial prediction to reduce the number of nodes and load on a single link. And the other is to directly upgrade the hardware, which increases the bandwidth capacity of the entire link

by expanding hardware boards and ports to reduce link utilization (traffic/bandwidth). Due to the shortcomings and limitations of the current scheme, it is difficult to improve the current unbalanced transmission network.

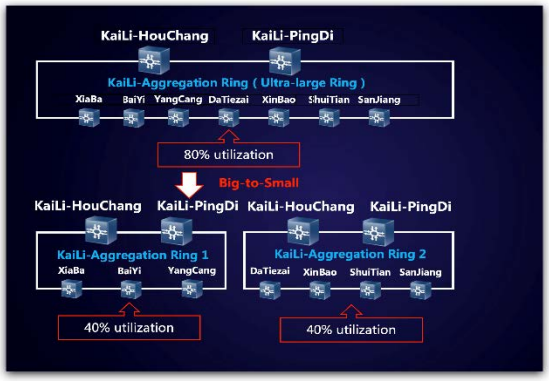


Fig. 2 – Solution1:Local(Ring) Adjustment: "big-to-small" optimization for some links, reducing the number of nodes and load

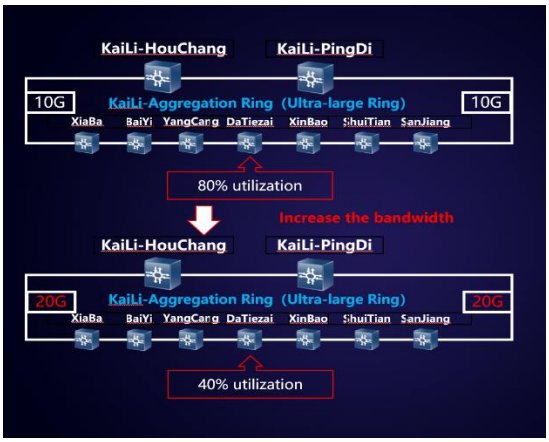


Fig. 3 – Solution2:Node expansion: Increase bandwidth of the link by upgrading hardware, and reduce link utilization

To solve this problem more effectively, the No Boundaries team has read a lot of literature, but there is no mature solution.

Qianyin Rao believes that applying the new SDN technology of 5G transmission network to transmission network can only solve the problems of delay and local optimization, and cannot support global topology optimization[1]. Intelligent control is still insufficient.

Lin Guo believes that wireless devices are the edges of networks, which differ from the nature of the transmission equipment[2] [3]. So a wireless network topology optimization scheme cannot be adopted.

Zhouwei Gang believes that the number of nodes in the data center and Internet scenarios is far less than that of the transmission network and cannot be applied[4] [5] [6].

Although no mature solution was found, the team was inspired to use machine learning instead of humans for topology optimization.

2. SOLUTION

2.1 Simulated environment

In order to explore the optimization of network topology based on machine learning, China Mobile selected more than 2,000 pieces of node topology information in three cities A, B, and C from the real network and the traffic situation of the nodes in the previous 20 days as the basic data of the contest, and the comprehensive evaluation indexes of topology optimization results (single day) are as follows:

$$\max \text{object\_ratio}_{\text{avg}} - (E_{\text{avg}} + E_{\text{min}} + E_{\text{max}}) - \alpha \cdot \text{sub\_ratio} - \beta \cdot \text{hang\_ratio}$$

$\text{object\_ratio}_{\text{avg}}$: the hourly daily average value of the ratio of link bandwidth utilization optimized to the target range;

E_{avg}: hourly daily average value of the network topology load balancing E value;

E_{min}: hourly daily minimum value of the network topology load balancing E value;

E_{max}: hourly and daily maximum value of the network topology load balancing E value;

sub_ratio: ratio of the number of nodes on the secondary link in each link to the number of nodes in the entire link (hourly daily average);

hang_ratio: the ratio of the number of nodes hanging in each link to the number of nodes in the entire link (hourly daily average);

α, β:Constant coefficients (α=0.02, β =0.05).

The specific algorithm of the evaluation index is as follows:

Target range of link bandwidth utilization:

$$[\frac{3}{5} \times \mu,\ \frac{7}{5} \times \mu]$$

The calculation method of the load balancing E value is as follows:

E value = variance of the fp value of all links in the network topology

$$\text{Link bandwidth utilization} f_P = \frac{\sum_i^{\text{nodes}} f_{X_i}}{A}$$

The μ in the upper and lower limits of the target range is the average value of bandwidth utilization of all links in the network topology.

In the f_P calculation formula, f_X is the flow value of the network element nodes in the link except the link head and tail nodes, and A is the maximum value of the A value of the other nodes in the link except the link head and tail nodes.

2.2 Architecture design

Through team analysis, Qian Deng found ITU's machine learning framework in the future network (mainly containing three components, ML sandbox system, ML pipeline subsystem and management subsystem), and believed that the ML pipeline subsystem met the needs of this competition.

The ML pipeline subsystem consists of 7 parts, but the data has been provided for this competition, and the optimization results are given in the form of a table and do not need to be directly connected to the equipment. Therefore, SRC, C, D, and SINK are not involved in the development. The result mainly consists of three parts: PP for data cleaning, M for topology restoration and traffic prediction[9], and P for optimization of the topology according to optimization rules and predicted traffic strategy.

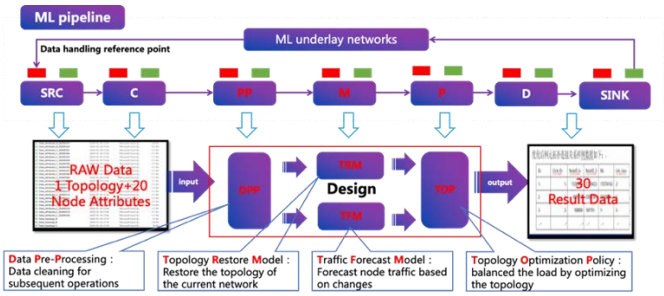


Fig. 4 – Mapping ITU architecture

2.3 Algorithm selection

Since the data comes from the real network, there are certain deficiencies, so the data preprocessing mainly considers two aspects of data integrity and ease of use:

Data integrity: Zezhong Feng uses pandas to check the integrity of key fields (traffic, latitude, longitude, connection relationship, etc.) and fill in missing data to ensure normal operation of subsequent predictions and optimizations.

Ease of use: Lin Xi believes that data is based on nodes. To restore the connection of nodes in the network topology, it is necessary to transform the data structure to facilitate subsequent calls. Compared with the adjacency table, using an adjacency matrix to store the connection relationships of nodes can improve query efficiency.

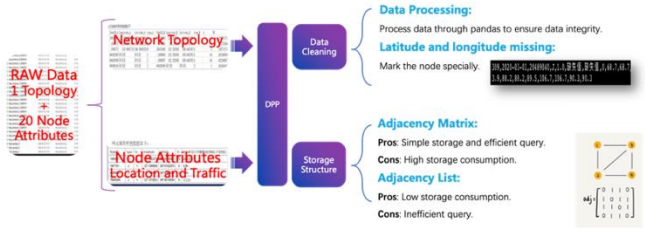


Fig. 5 – Node structure

2.4 Modeling

Regarding the Topology Restoration Model (TRM) and Traffic Forecast Model (TFM), Zhouwei Gang believes that the essence of topology restoration is to organize and form a new data set according to the specified conditions from the original data set. Therefore, search algorithms can be used for processing. Traffic forecasting is based on the changes of things in the past, mining the law of change, predicting the future, and there is a strong correlation with time, so the time series prediction algorithm is used for processing [10][11].

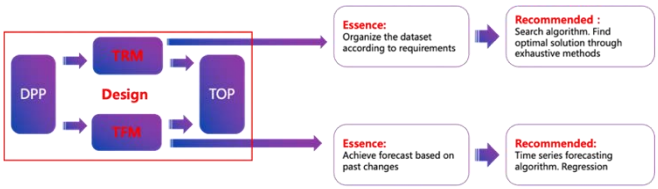


Fig. 6 – Time series prediction algorithm

Before TRM, Data Preprocessing (DPP) had converted the node connection relationship into a graph representation. Therefore, the process of finding a set of links in the topology that meets the specified conditions (topology restoration or topology optimization) is essentially a graph search problem. The most commonly used search algorithms for graphs are Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms. Lin Xi said that this project has certain requirements for convergence time, so the breadth-first search algorithm is adopted for higher search efficiency.

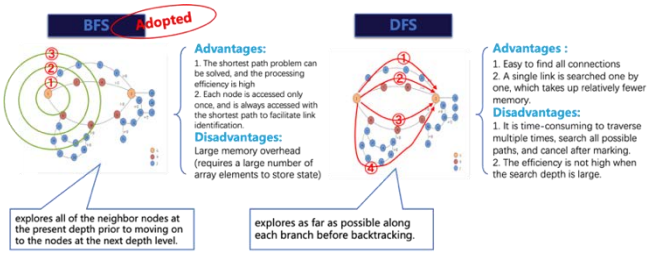


Fig. 7 – BFS and DFS

However, traditional BFS cannot visit the visited node, and there is a situation in which the node is visited multiple times in the simulation data. To solve this problem, Lin Xi modified the BFS search status part so that the improved algorithm can repeatedly visit the node and adapt the search of the link set.

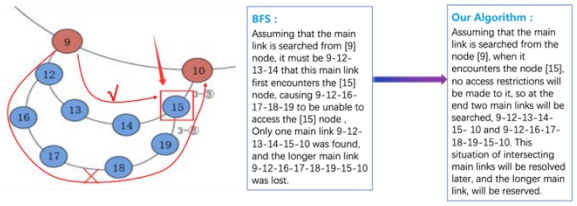


Fig. 8 – Improved BFS

For TFM,mainstream time series prediction algorithms include multiple linear regression, Autoregressive differential Moving Average (ARIMA) and Long-Short-Term Memory network (LSTM) model [7]. By analyzing the characteristics and data of this research, the influence between the data is almost non-linear. Among the above prediction models, the LSTM model is designed with a special structure to memorize and filter the changes in traffic on the time scale. Therefore, we choose the LSTM recurrent neural network model for prediction to meet the requirements of this prediction scenario.

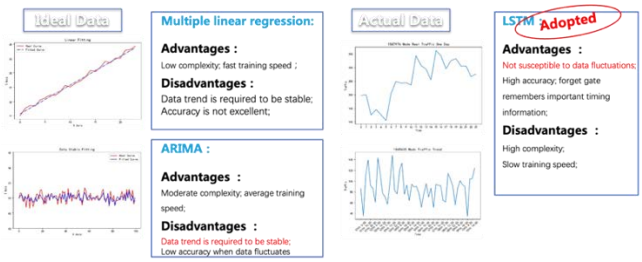


Fig. 9 – LTSM adopted

Zezhong Feng believes that the bottleneck for improving the accuracy of traffic forecasting is that there are few samples and few influencing factors, many nodes of different types, so the model is in two layers to reduce overfitting and under-fitting. We build a model for each node and increase the flow information carried by each neuron from 1 to 24, so that the model can fit more flow changes. In the end, the average accuracy of TFM increased from 91.7% to 95.3%.

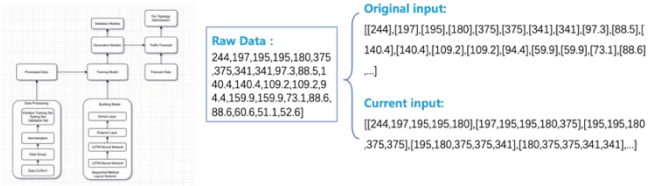


Fig. 10 – Improved input

In order to further improve the prediction accuracy, Zezhong Feng tried to add more features to improve the traffic forecast model. In the time dimension, we add the two features of the day's weather and the weekend to make it have traffic and weekend features; in the space dimension, we add the node traffic at a certain distance around the node as a feature, and we analyze that it is within a certain range (tentative 2 Km) where nodes have an impact on the node traffic of the current link, and such node traffic is added to features.

Zezhong Feng's experimental results indicate that:

By adding weather features, the average forecast accuracy of the model is increased by 0.4%;

by adding the features of working day/off day, the average forecast accuracy of the model is increased by 1.6%;

by adding the traffic chrematistics of surrounding nodes, the average forecast accuracy of the model is increased by 1.8%;

after all three features were added, the average forecast accuracy of the model increased from 91.7% to 95.3%.

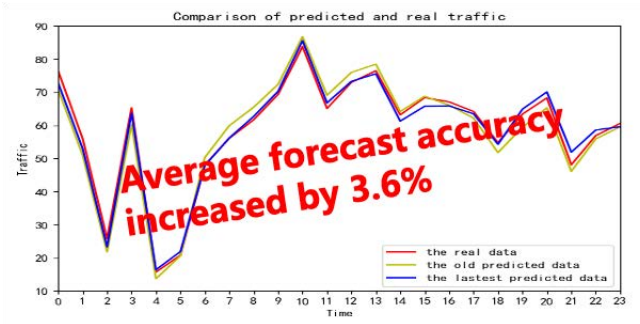


Fig. 11

2.5 Optimization strategy

In terms of Topology Optimization Strategy (TOP), Zhouwei Gang proposed an optimization strategy of "(ultra-low)*3". First calculate the utilization rate of all links and the average value of the entire network, and set a threshold to divide all links into three categories: overload, low load and normal. The overload links are processed first, and then the low-load links are processed. After finishing, adjust the threshold value, do it twice again, and finally obtain the optimized topology by completing three threshold adjustments.

Fig. 12 – Topology optimization strategy

Zhouwei Gang completed the optimization rules for overload and low-load links, and Lin Xi realized the aggregation of low-load links with the same head and tail nodes.

Overload link optimization scheme：

Calculate the utilization rate of all links and the average value of the entire network based on the predicted node traffic, search for all overloaded links to form a set, and select a link, as shown in the red link on the left. Along the link sequence, each node is judged according to rules, whether it can establish a link with other link nodes, and the results are formed into a set. According to the load of the original link and the load of the new link, find a pair of nodes from the set, disconnect them, and connect them to other links to form a new link.

The low-load link optimization rules are similar and will not be elaborated.

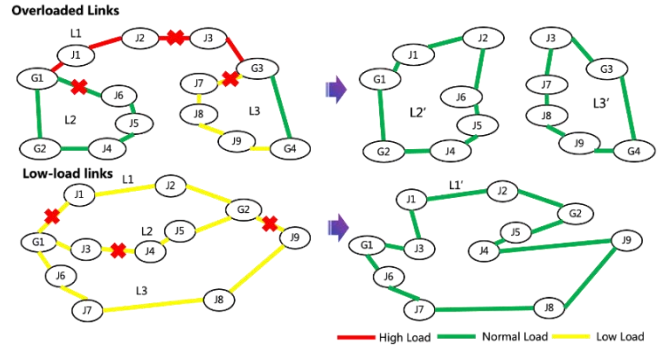


Fig. 13 – Link adjustment

L stands for link, J stands for J node. Through the "(ultra low)*3" TOP (optimization strategy refers to the calculation of three operations repeated three times):
1. Optimize overloaded links
2. Optimize low-load links
3. Adjust the threshold and recalculate the overload link and the low-load link

The algorithm can complete topology optimization in 55.5 minutes. Lin Guo used Qunee (Web graphics component solution) to complete the topology map. The left picture of the figure below is the original topology of City C, and the right picture is the optimized topology. Green nodes indicate adjusted nodes.

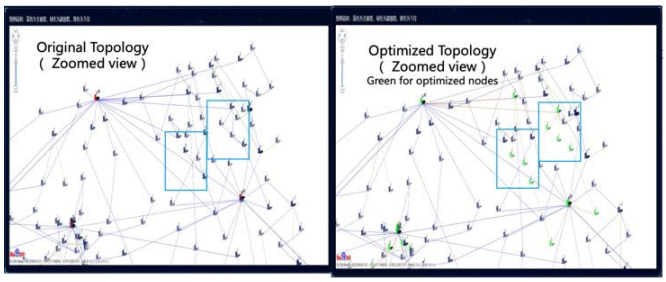


Fig. 14 – Node load reduction

But the "(ultra-low) *3" TOP (topology optimization strategy) is still the topology optimization of the link granularity, can it be further refined to the node granularity?

Lin Xi introduced a greedy algorithm to improve the optimization strategy. The core strategy is to refer to the index score when optimizing overloaded links and low-load links, and always move the nodes to get the highest index score. In this way, the node is moved to the optimal link.

The calculation of the index score is mainly related to the link optimization ratio, the average, minimum, and maximum value of link bandwidth utilization, and the ratio of secondary links and downstream nodes in the link. Using the optimization strategy of the greedy algorithm, the nodes are not randomly moved to other links, but selectively moved to the links that make the index score higher, and the optimization granularity is refined from the link level to the node level, thereby improving the overall optimization effect, with the link optimization ratio increased by 0.4. Although the calculation time doubled, it did not exceed 15 minutes, and the optimization evaluation score increased by about 10%.

At the same time, Lin Xi has tested the genetic algorithm. For the topic of network topology optimization, the genetic algorithm involves more debugging such as parameter coding, selection, crossover, mutation, etc. For example, topology optimization is difficult to map and transform in the parameter coding part, and it involves rule constraints during crossover, considering the entire genetic algorithm for topology optimization is actually very complicated, finally only the greedy algorithm is used.

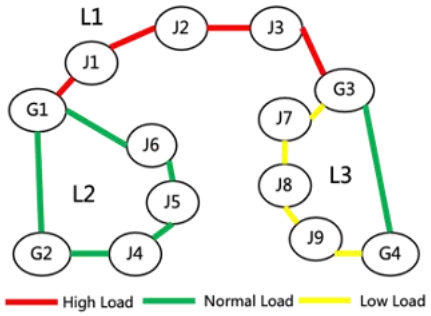


Fig. 15 – Greedy algorithm

	J1	J2	J3	J4	·····
L1	0.4618	0.5026	0.5185	0.5531	·····
L2	0.4767	0.5345	0.5567	0.5732	·····
L3	0.5068	0.5123	0.5234	0.5613	·····

The link optimization ratio increased 11%

Fig. 16 – Increased 11%

2.6 Real-world scenario

Qianyin Rao introduced the results into Kaili City, with 913 chains composed of 3191 network elements, and an AI optimization algorithm output of 60 high-load solutions, of which 29 have passed the expert review and are qualified for implementation. In order to ensure safety, the operation is carried out in batches. Each optimization does not exceed 50 nodes, and the scope of influence is controlled within 10 links. After the operation is completed, observe the stability of the network for 3 days before proceeding to the next operation. Up to now, 11 batches of adjustments have been completed, and 16 high-load links have restored reasonable loads.

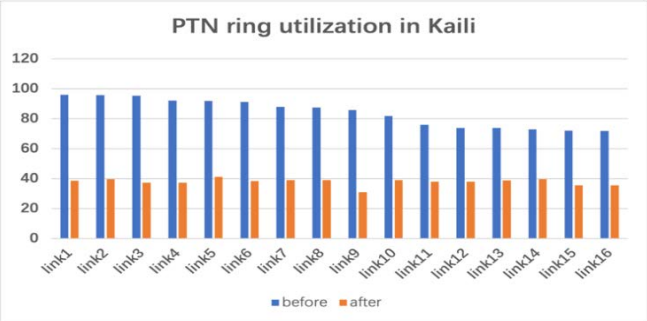


Fig. 17 – PTN ring utilization

For these 16 overloaded links, compared with the traditional solution, this network topology optimization saves construction costs by 2.6 million RMB, reduces personnel investment by 17 man-days, and increases network absorption traffic by 79TB/day.

Plan	Traditional Plan	AI Plan
Construction Cost	¥ 3.05M	¥ 0.45M
Human Input	21 Man-Days	4 Man-Days
Low-load links	40	3
High-load links	5	0
Capacity (day)	102T	181T

Fig. 18 – Economic performance

2.7 Standard packaging and rapid promotion

Lin Guo said, 'In order to meet the needs of rapid replication and promotion, according to ITU ML architecture and recommendations, the requirements in ITU recommendations are implemented by docker and Kubernetes'. Docker container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another K8s, is an open-source system for automating deployment, scaling, and management of containerized applications. According to the test, the development time is shortened by 70%, data sharing is applied to reduce 50% storage, and fault self-healing is realized. Finally, the overall architecture is completed on the basis of integration with ITU architecture.

3. RESULTS AND DISCUSSION

3.1. Results

In accordance with the ITU standard Lin Guo evaluated the results, the application results reached L3 in analysis, decision and demand mapping, and the data collection reached L2. Since the newly-built link in the transmission topology optimization involves physical entity operations and cannot be fully implemented by the software system, the action implementation only reaches L1.

But the team believes that after the results continue to improve, the three parts of data collection, analysis and decision can be upgraded from the current L2 and L3 levels to L5. The demand mapping team believes that it needs to accumulate more expert experience to break through to L4. If the results are applied to other similar network topology optimizations that do not require physical adjustment, the team believes that L5 can be achieved.

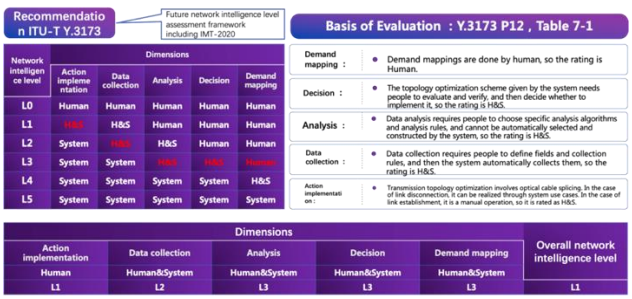


Fig. 19 – Intelligent level

At the same time, the team believes that there is still room for further evolution of this result.

First of all, continuing to optimize the algorithm is a good choice. At present, we only use the greedy algorithm, but there are actually many optimization algorithms that have not been tested and verified for similar problems.

Secondly, conduct pilot projects in more locations to accumulate sample data to adjust and optimize strategies to improve accuracy.

Finally, after the topology adjustment is completed, perform a certain period of observation, roll back, and analyze the reasons for those that have not reached expectations, and guide the improvement of the strategy.

3.2. Discussion

Through this research, with the support of ITU planning, our results have achieved breakthrough success. Compared with traditional operation and maintenance methods, we have obvious advantages. We think that this AI/ML application in the communications industry has positive significance in three aspects:

The communication network before 5G mainly communicates between people, and the three typical scenarios in the 5G standard reflect that 5G has increased the communication between people and things, and between things. This type of communication will not only change the way of human life, but also change the mode of social operations. Facing this great prospect, ITU, global operators and equipment vendors are all vigorously promoting the construction and operation of 5G networks. But this also causes the complexity of the 5G network to far exceed the previous network, and the construction cost and operation and maintenance cost have doubled. The results of this combination of communication operation and maintenance and AI can greatly reduce the cost of 5G network construction and operation and maintenance, and accelerate the promotion of 5G. Therefore, it is recommended that the ITU continues to improve the integration of AI/ML technologies in the communication specifications to provide support for global operators and equipment vendors, so as to burst more intelligent operation and maintenance results, promote 5G networks to change society as soon as possible, and enable us to enter a better future.

Fig. 20 – 5G for everything

In China Mobile, an intelligent operation and maintenance ecosystem of developers + platform + standards have been formed. The "Nine days" intelligent intermediate platform created by China Mobile provides functions for developers and gathers the wisdom of developers who work in accordance with ITU standards. The result is screened by China Mobile and then promoted, forming an endless ecosystem. Relying on this ecosystem, China Mobile has cultivated many intelligent operation and maintenance results, supports China Mobile in building the world's largest 5G network, and constantly discovers new problems in construction, operation and maintenance, and new problems have become new goals for developers. In this way, we can realize the efficient operation and maintenance of complex networks.

Fig. 21 – ecological environment

Most of the current AI development is the to customer mode that focuses on serving users, based on user needs, and conforms to users' living habits. Based on the ITU specifications, this project has achieved the integration of the communications industry and AI. This is an exploration of the to business model of AI development serving the industry. It is developed based on industry needs and meets industry development specifications. It is foreseeable that more and more AI services will adopt the industry-to-B model in the future, this research will provide an example for them.

Fig. 22 – AI to C and AI to B

REFERENCES

[1] Rusek, K., Suárez-Varela, J., Mestres, A., Barlet-Ros, P., & Cabellos-Aparicio, A. (2019, April). Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN. In Proceedings of the 2019 ACM Symposium on SDN Research (pp. 140-151).

[2] Gui, J., & Zhou, K. (2016). Flexible adjustments between energy and capacity for topology control in heterogeneous wireless multi-hop networks. Journal of Network and Systems Management, 24(4), 789-812.

[3] Ma, R., Cao, X., Zhang, S., Liu, L., Cheng, Y., & Sun, C. (2018, December). A DBN-Based Independent Set Learning Algorithm for Capacity Optimization in Wireless Networks. In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[4] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., & McKeown, N. (2010, April). Elastictree: Saving energy in data center networks. In Nsdi (Vol. 10, pp. 249-264).

[5] Zhuo, D., Ghobadi, M., Mahajan, R., Förster, K. T., Krishnamurthy, A., & Anderson, T. (2017, August). Understanding and mitigating packet corruption in data center networks. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (pp. 362-375).

[6] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).

[7] F.A. Gers, J. Schmidhuber and F. Cummins, Learning to forget: Continual prediction with lstm, 1999.

[8] Qing He, Arash Moayyedi, György Dán, Georgios P. Koudouridis, Per Tengkvist, "A Meta-Learning Scheme for Adaptive Short-Term Network Traffic Prediction", Selected Areas in Communications IEEE Journal on, vol. 38, no. 10, pp. 2271-2283, 2020.

[9] Y. Tian and L. Pan, "Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network", 2015 IEEE InternationalConference on Smart City, pp. 153-158, 2015.

[10] M. Ball and R. M. Slyke, "Backtracking algorithms for network reliability analysis", Ann. Discrete Mathematics, vol. 1, pp. 49-64, 1977.

[11] R. R. Boorstyn and H. Frank, "Large-scale network topological optimization", IEEE Trans. Communications, vol. COM-25, no. 1, pp. 29-47, 1977.

[12] ITU-T, "Architectural framework for machine learning in future networks including IMT-2020", ITU white paper. Communications, Rec. ITU-T Y.3172, pp. 14-19, 2019.

AUTHORS

ZhouWei Gang obtained a master's degree in communication engineering in 2012. He worked for China Mobile Guizhou Company, mainly engaged in network analysis, data processing, IT development and artificial intelligence research. He has published 8 papers (in Chinese) including "Research on VoLTE MOS Value Slicing Evaluation Method", "Supporting 4G Business Development Based on the "User-Network-Business" Three-dimensional Method" (Chinese), and published "A Language Conversion Method Based on Artificial Intelligence". He has been involved in five patents including a method and device for early warning and judgment of single-point operation of transmission services. At present, he mainly studies the application of artificial intelligence in the field of the telecom industry.

Qianyin Rao obtained a bachelor's degree in the telecom industry engineering in 2005, and worked in the field of China Mobile transmission and IP bearer network production management and optimization. He published a patent "A method of MPLS service application based on VPN nesting technology" and "A degradation of IP network link quality" "Methods of Detection and Protection", won 1 group-level and 3 provincial-level scientific and technological innovation awards in the field of transmission and bearing. At present, his main research is the automatic detection of the same route of the transmission network and automatic service opening.

Lin Guo received a bachelor's degree in engineering in 2005 and worked in China Mobile in charge of core network maintenance, independent research and development, and AI design. He has published 7 national patents, "A Smart Remote Video Interaction System Based on AR", "A New Method for Visual Fiber Cable Fault Location and Presentation", "A method for locating the root cause of communication network equipment in the network management system. Methods" etc. The application of 3D modeling technology in wireless is currently being studied.

Ze Zhong Feng holds a Bachelor of Engineering degree and is a senior engineer of China Mobile Guizhou company, engaged in AI research, time series prediction, and is committed to algorithm optimization of network topology. He is also committed to big data, machine learning and deep learning.

Lin XI, received a bachelor's degree in software engineering in 2018. Currently he is working in Guizhou Mobile, responsible for IT R&D and AI capability research. Currently the main research is structured data mining.

Qian Deng received a bachelor's degree in 2017 in information and computing science and a bachelor's degree in software engineering. He has worked in the complex IT R&D and AI capability research field of China Mobile. He has published the paper "The Realization of A Penetration Testing Assistant Tool", and published the patent "A Language Recognition, Method, System and Device". Currently, he mainly researches the direction of data mining through machine learning.

NOTE: The patent titles of the above-mentioned papers are all translated in Chinese and published in Mainland China.

ANALYSIS ON ROUTE INFORMATION FAILURE IN IP CORE NETWORKS BY NFV-BASED TEST ENVIRONMENT

Xia Fei[1], Aerman Tuerxun[1], Jiaxing Lu[1], Ping Du[1], Akihiro Nakao[1]
[1]The University of Tokyo, Japan

NOTE: Corresponding author: Ping Du, duping@g.ecc.u-tokyo.ac.jp

Abstract – *Stable and high-quality Internet connectivity is mandatory for 5G mobile networks. However, the pandemic of COVID-19 has forced global and large-scale staying at home and telecommuting in many countries. The increasing traffic has induced more pressure on networks, devices and cloud data centers. It becomes an essential task for network operators to enable their ability to automatically and rapidly detect network and device failures. We propose a highly practical method based on highly practical technology. Our method has a high generalization ability that can efficiently extract features from large-scale unstructured data and ensure high accuracy prediction. First, 997 useful features are extracted from 28GB-per-day network logs. Then, a differential approach is employed to preprocess the extracted features so as to highlight the differences between normal and abnormal states. Third, those features are refined based on the feature importance we calculated. According to our experiment, the proposed feature extraction and refinement method can reduce computation without degrading the performance. Among the five types of failures, we achieve a 100% recall rate in four types and the rest can also reach 71%. Overall, the total average prediction accuracy of the proposed method is 94%.*

Keywords – Core network, failure detection, route information, machine learning

1. INTRODUCTION

The pandemic of COVID-19 has forced global and large-scale staying at home and telecommuting in many countries. The implementation of social restrictions increases Internet traffic, particularly the traffic of remote working, web meetings, and online education. For instance, Netflix has faced a surge in subscriber numbers, with almost 16 million people signing up for accounts in the first three months of 2020 [1]. Zoom's daily active users spiked to 200 million in March 2020, up from 10 million in December 2019 [2]. Such increasing network traffic has induced more network and device failures than before. For example, it is reported that Google has suffered an estimated $1.7M loss in advertisement revenues during their "outage" in December 2020 [3]. Thus, how to automatically and rapidly detect network and device failures has become an essential problem in daily operation.

Network technologies such as 5G, have dramatically changed the telecommunication environment that brings faster speed experience to us. 5G mobile networks require stable, high-quality Internet connectivity, but when a failure happens, the consequences of that failure are extremely serious. In addition, since the Internet is operated mutually among ISPs, even if a failure occurs in a certain ISP domain, the failure spreads rapidly all over the world. However, only experienced ISPs can deal with such a network failure that affects the world. It is desirable that anomaly detection could be performed automatically and promptly. The IP backbone network of one ISP is interconnected with others via Border Gateway Protocol(BGP) routers. A BGP router needs to continuously update the route information from the received internal/external route information and provide appropriate feedback. Thus, these BGP routers play very important roles in 5G services, and in order to maintain a certain level of service, it is desirable to immediately detect hardware and software defects and malfunctions. Moreover, increasing network traffic also brings challenges to data-based optimization. Recent hot AI technologies provide novel approaches that are able to migrate our focus of work from fault handling to fault prediction, which allows operators to take precautions in advance.

Based on the data sets [4] provided by KDDI Corporation, we propose an efficient method to predict network and device failures from large amounts of unstructured log files in real time. Our proposal contains three main steps: *Feature Extraction*, *Feature Refinement*, and *Feature Reduction*. In the Feature Extraction step, we extract 997 features from 28GB per day of unstructured log files, and merge tagged features from the following three kinds of JSON log files: *physical-infrastructure*, *virtual-infrastructure*, and *network-devices*. As for the BGP-related entries, we use the number of next-hops in each array and corresponding prefixes as features.

In order to derive metrics that are changed when there is an the occurrence of a failure, we highlight the difference between normal and abnormal entries and define a new feature named Differential Data to reflect the variations between abnormal data and normal data. After the data processing, one CSV file that could be utilized for training or evaluation is generated.

For the sake of importance analysis, the XGBoost [5] model is trained by us to calculate the importance scores which work as the reference in the *Feature Reduction*

phase via observing the changes in accuracy which are trained by different numbers of features, we have two observations: (1) The highest accuracy is 94% when the number of features is more than 150; (2) Accuracy could achieve 93% if we use only the top 30 most important features, without obvious performance degradation.
According to our evaluation, we achieve a 100% recall rate when detecting the following four network and device failures: *Node-Down*, *Interface-Down*, *Ixnetwork-BGP-Injection*, and *packet loss & delay*. There is a 71% recall rate of Ixnetwork-BGP-Hijacking detection, while the total average accuracy of our proposal is 94%. XGBoost, Random Forest, and LightGBM [6] have been demonstrated in our experiments that they outperform other methods in terms of training and inference time.
In summary, the main contributions in this paper are as follows.

- First, we define a staged method including feature extraction from unstructured network logs, a differential approach to highlight the differences between normal and abnormal states and several ML models to realize failure classification.
- Then, we apply the staged method to six popular machine learning algorithms, including Decision Tree (DT), XGBoost, LightGBM, Multilayer Perceptron (MLP), Random Forest (RF), and Support Vector Machine (SVM). After a comparative evaluation, we reveal that the tree-based models (such as XGBoost) outperform others in detecting network failures.
- Third, we employ a model refinement method to sort the features according to their importance score. We confirm that with the most useful features we gain computational speed without obvious degradation of accuracy.
- Finally, we also find that latency and loss are confused according to the RF confusion matrix so that they are hard to predict inherently.

The rest of this paper is structured as follows. Section 2 describes the relevant research on network fault analysis. In Section 3, we present our extraction method from raw data and comparative analysis of ML-based faults classification. Section 4 shows the experimental results obtained using our method and the evaluation of comparison results. Finally, we provide a brief conclusion in Section 5.

2. RELATED WORK

There has been numerous literature concerning network faults detection. Most approaches rely on predefined rules, thresholds, and expert experiments. Mitchell et al. present a fault detection system for LAN networks [7]. The system is based on a set of rules defined on the data collected from the network monitoring process and the expertise of the network administrators. Although these methods can be realized automatically through scripts, they are usually low inefficiency and high in human labor costs [8, 9]. Therefore, approaches including Finite State Machine (FSM) and probabilistic approaches have also been researched [10–12]. Authors in [10] propose an FSM-based model and realize fault detection of partially observed data sequences. With the aid of FSM, [11] employ a probability approach to choose to synchronize conditions and optimally develop adaptive strategies. However, these traditional methods can hardly handle the frequent and dynamic changes in the network topology. On the other hand, the data volume obtained from managed entities is increasingly large in the era of 5G, and huge benefits can be leveraged from data-driven fault detection methods.
With the spread of the usage of Machine Learning (ML) technology in many fields, more and more studies have been proposed on network fault analysis using ML. Networking itself can also benefit from this promising technology. İF Kilinçer et al. propose a Bayesian method for monitoring and diagnosing faults that may occur in the Internet line [13]. They extract data via edge switching devices in a network campus area and use the Bayesian method to classify. It has been found that the accuracy of the classification results is over 90%. Ruiz et al. propose a probabilistic failure localization algorithm based on Bayesian Networks (BN) to localize and to identify the most probable cause of failures impacting a given service [14]. The authors use time-series monitoring data extracted from several light paths. When a service detects excessive errors, an algorithm uses the trained BN to localize and identify the most probable cause of the errors at the optical layer. Sauvanaud et al. propose anomaly detection and root cause localization for VNF using a supervised machine learning algorithm [15]. This approach detects Service Level Agreements'(SLA) violations based on monitoring data. It can pinpoint the root anomalous VNF VM causing SLA violations and achieve high recall, high precision, and low false alarm rate. Their experiments in [13, 14], and [15] show that the proposed algorithm can achieve high accuracy of fault classification. However, they do not compare their method with multiple ML algorithms or other training conditions.
Srinikethan et al. compare three ML algorithms that include SVM, MLP, and RF performance in terms of their link fault detection [16]. The authors develop a three-stage Machine Learning-based technique for Link Fault Identification and Localization (ML-LFIL) by analyzing the measurements captured from the usual traffic flows, including aggregate flow rate, end-to-end delay, and packet loss. Stadler et al. propose a method to predict service-level metrics from network device statistics using ML [17]. The authors adopt a work-regression tree and RF and investigate their prediction performance. They also compare the performance under several training conditions. References [16] and [17] compare the performance of multiple ML algorithms and seek to ascertain the effect of training conditions. However, their ML model's goal is fault detection and predictive service metrics, and it does not cover enough fault classification.

Qader et al. compare the performance of faults classification using K-Means, Fuzzy C Means (FCM), and Expectation-Maximization (EM) [18]. They use data sets obtained from a network with heavy and light traffic scenarios in the routers and servers and build a prototype to demonstrate the network traffic faults classification under given scenarios. The results show that FCM could achieve higher accuracy than K-Means and EM. However, it requires more time to process data. The authors focus on the data related to the physical interface only. Thus there is insufficient research on faults classification in an Network Function Virtualization(NFV) environment.
Recently, KDDI presented an ML comparison framework for network analysis [4]. It includes four functional blocks: data set generator, preprocessor, ML-based fault classifier, and evaluator. The data set generator can periodically generate failure data, which can be used in the ML-based fault classification task. They use three algorithms [Multilayer Perceptron (MLP), Random Forest (RF), Support Vector Machine (SVM)] to train and evaluate. The result shows that RF provides the highest performance even with a small amount of data, and SVM could improve its performance by increasing training data, feature reduction, or balance adjustment of normal/abnormal samples. However, the feature extraction method and training efficiency are not mentioned in their study. Training efficiency is an important metric for the evaluation of training models. Feature extraction is an essential step in achieving the excellent performance of an ML method. Especially for a large amount of network log data, efficiently extracting useful information from raw data can allow our model to perform better in a much shorter training time.

3. METHODOLOGY

This section introduces the data sets and shows how we extract features. Then we introduce several machine learning models used in this research.

3.1 Data preprocessing

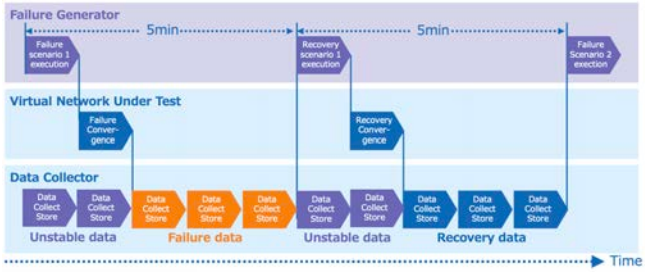


Fig. 1 – Data collection principles [4].

As shown in Fig. 1, The data sets used for this study are created in the NFV-based test environment simulated for a commercial IP core network. In this sense, synthetic data is similar to real data, resulting from the NFV-based test environment. The data sets consist of labels of normal/abnormal traffic, performance monitoring data sets such as traffic volume and CPU/Memory usage ratio, and route information such as Border Gateway Protocols (BGPs) static metrics and BGP route information.
The data collector from KDDI collects and stores data sets every minute from the network. Once a failure is intentionally caused or recovered, the network indicates a failure or normal status after a transition period, corresponding to failure data (orange arrows) and recovery data (blue arrows).
The time interval between a failure and a recovery is 5 minutes (Fig. 1). The data sets for training and evaluation provided by KDDI include four types, as in Table 1, which are *Label-Failure Management*, *Virtual-Infrastructure*, *Physical-Infrastructure* and *Network-Device*.
The training data set consists of 8 days of data, totaling approximately 120G JSON files. The evaluation data set consists of 7 days of data, totaling about 100G of JSON files.

Table 1 – Four types of data sets for learning and evaluation

Category	File Name	Data Format
Label	Label-Failure Management	json
Log Data	Virtual-Infrastructure	json
Log Data	Physical-Infrastructure	json
Log Data	Network-Device	json

3.1.1 *Data collection and merging method*

The JSON file's content is enormous, and most of the information is useless string description information. So we iterate through each object, looking for objects of numeric type. We extract these objects as features from log files (in JSON format) and merge them with labels into a CSV file based on time (Fig. 2).
We utilize paths like "key1/key1-1/key1-1-1..." as keys to extract features from physical, virtual, and network JSON log files for all log files. For BGP-related entries, we use the number of next-hops in each array and their prefixes as features.

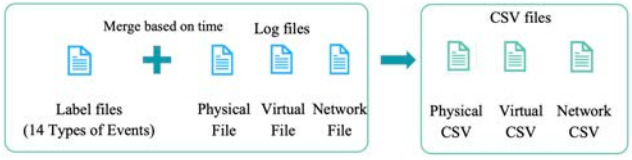


Fig. 2 – Data mergence principles

3.1.2 *Data differential method*

This subsection explains how our comparison framework preprocesses Performance Management (PM) data to put into Machine Learning (ML) models for training.
As shown in Fig. 3, each failure generation cycle is 5min. In the failure generation cycle, the last-minute data in the previous cycle is considered as regular data, and the last-minute data in the current cycle is considered as failure data. To highlight the differences between normal and

abnormal data entries to derive metrics that have changed since a failure, the differential data between the abnormal data and normal data is used as input features. After that, we can get three types of files in the data set, which are physical, virtual, and networks. To train a unified model for diverse network events, we merge all data sets into one CSV file for putting into ML algorithms. The process is shown in Fig. 4. Finally, the data set for training consists of 930 lines with 996 features, and for evaluation consists of 840 lines with 996 features.

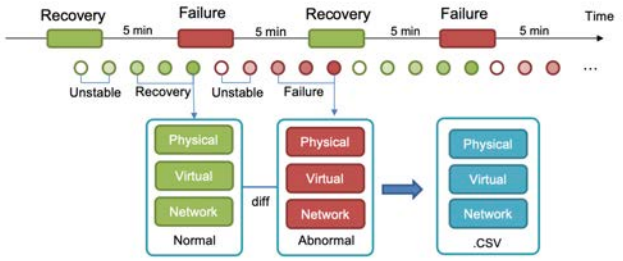


Fig. 3 – Data differential method

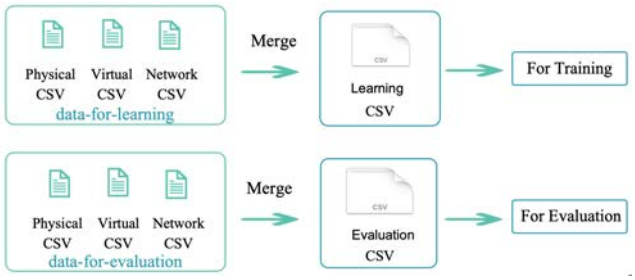


Fig. 4 – Data merging method

3.1.3 Label description

As shown in Table 2, we have five categories of labels for prediction, which are *Type1: node-down*, *Type3: interface-down*, *Type57: tap-loss (delay)*, *Type9: ixnetwork-bgp-injection*, and *Type11: ixnetwork-bgp-hijacking*.

3.2 Machine learning methods

In the related work in [4], Multiplelayer Perceptron (MLP), Support Vector Machine (SVM), and Random Forest (RF) are employed. In this study, as an extension of the related work, three other kinds of tree-based models, Decision Tree (DT), XGBoost (XGB), and LightGBM(LGBM) are also utilized.

3.2.1 Multiplelayer Perceptron (MLP)

MLP is a feed-forward artificial neural network that maps input data to the appropriate output. An MLP is a network of simple neurons called a perceptron which computes a single output from multiple real-valued inputs. A Perceptron forms a linear combination to its input weights and puts the output through a nonlinear activation function. The mathematical representation of MLP output is:

$$y = \varphi(\Sigma_{i=1}^{n} w_i x_i + b) = \varphi(W^T X + b)$$

where W is the vector of weights, X is the vector of inputs, b is the bias, and φ is the activation function.

As a neural network based model, the MLP algorithm is a general function approximation method that can fit complex functions and adequately approximate complex nonlinear relationships. It has a wide range of applications and has features of high accuracy. It is often used to solve classification problems. However, neural networks require manual determination of a large number of parameters, such as network topology, initial values of weights, and thresholds. Learning may be not sufficient when the parameter selection is inappropriate, and it is easy to fall into local extremes. Besides, since MLP is a black-box process, the learning process cannot be observed, and the output is difficult to interpret, which can affect the credibility and acceptability of the results.

3.2.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a linear machine working in a high dimensional feature space. SVM employs a nonlinear mapping to map the N-dimensional input vector x into a K-dimensional feature space (K>N). The problem that SVM tries to solve is to find an optimal hyperplane that correctly classifies data points by separating the points of two classes as much as possible. Both classification and regression tasks transform the learning task into a quadratic problem, but the way of creating SVM networks varies depending on the classification and regression tasks [19, 20]. Excellent introductions to SVM can be found in [21].

The main advantages of SVM are (1) able to work with high-dimensional data; (2) high generalization performance without the need to add prior knowledge, even when the dimension of the input space is very high.

Compared to MLP, SVM performs better in classification mode. And in regression mode, MLP has better generalization ability. In most cases, the observed performance difference is negligible [22].

3.2.3 Decision Tree (DT)

A decision tree is a supervised machine learning algorithm that can be applied to both classification and regression problems. Usually, it is top-down tree-like structures that explain the decision-making rules for prediction. The node from where the tree starts is known as a root node. The node where the tree ends is called the leaf node. Each internal node can have two or more branches. A node represents a particular characteristic, while a branch represents a range of values. These ranges of values act as partition points for the set of values of the given characteristic. In Fig. 5, we provide an illustration of a decision tree, which is also used in our experiments:

Table 2 – Five categories of labels for prediction

Scenario	TypeNo.	Type Name	Description
Network Element(NE) Failure	1	Node Down	Unplanned reboot of a NE
Interface Failure	3	Interface Down	Cause an interface down
Interface Failure	57	Packet Loss and Delay	Cause the packet loss and delay on an interface
Route Information Failure	9	BGP Injection	Inject the anomaly route from another SP
Route Information Failure	11	BGP Hijack	Hijack the own origin route by another SP

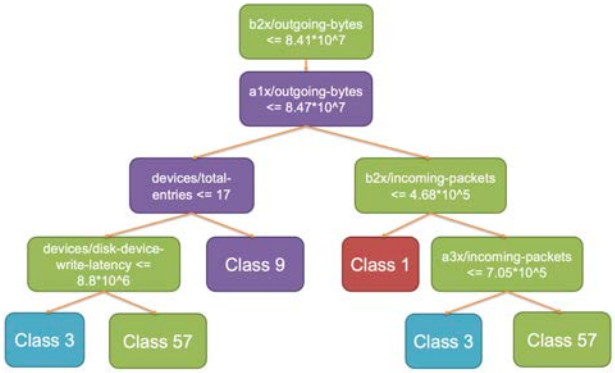


Fig. 5 – Visualization of a decision tree

A decision tree is a very intuitive data structure. Visualizing a decision tree can give valuable insights into the model's learning and how domain-relevant the learning is. In addition, the Decision Tree is fast and performs well on large data sets. It is also unaffected by outliers.
However, there are some shortcomings of a Decision Tree including:

- A decision tree might lead to overfitting when a tree is very deep. As the decision to split the nodes proceeds, each attribute is taken into consideration. It will try its best to fit all the training data accurately, which leads it to learn too much about the features of the training data and lose its ability of generalization.
- A decision tree is greedy and tends to find the local optimal solution instead of considering the global optimal solution. At each step, it uses some technique to find the best node. However, the local best node may not be the global best node.

3.2.4 Random Forest (RF)

To overcome the shortcomings of DT, random forest comes to the rescue. It was first proposed by Tin Kam Ho in 1995 [23]. Random forest consists of a large number of individual decision tree that operate as an ensemble. In the training process of RF, each individual tree in the RF spits out a class prediction and the class with the most votes becomes our model's prediction (Fig. 6). Random forest models are so effective because a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

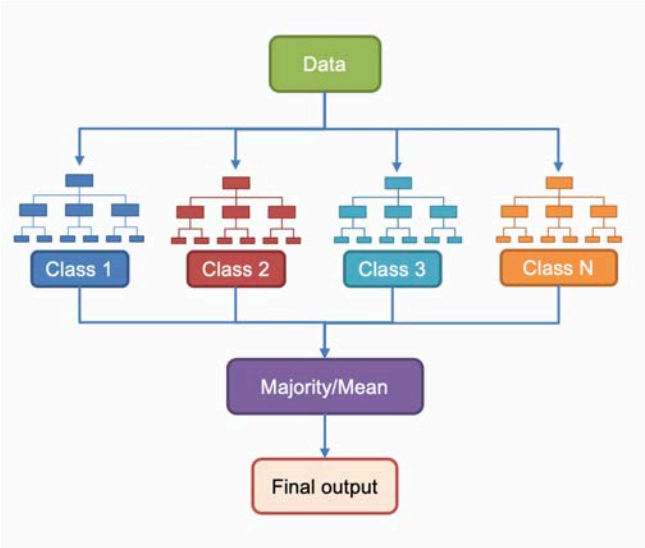


Fig. 6 – Visualization of an RF model making a prediction

Different with the decision tree model, which employs all the given data to determine the classification rules, a random forest classifier can randomly sample from the given data and build more than one decision trees. By considering the classification results of the several decision trees employing different subsets of the original data, the random forest method can effectively avoid overfitting. However, directly proportional to the model complexity, the training cost of random forest model would be higher than the decision tree model.

3.2.5 XGBoost (XGB)

XGBoost (XGB) is a successful machine learning model based on a gradient boosting algorithm proposed by Tianqi Chen [5]. Like random forests, gradient boosting is a set of decision trees. The two main differences are:

- Gradient boosting builds one tree at a time, while random forests builds each tree independently.
- Gradient boosting combines the results along the way, while random forests combines the results at the end of the process.

XGBoost stands for extreme gradient boosting. It is a specific implementation of the gradient boosting method that delivers more accurate approximations using the second-order derivative of the loss function, advanced regularization, and parallel computing.

The objective of XGBoost is to not only prevent overfitting but also optimize the computational resources. This is obtained by simplifying the objective functions that combine predictive and regularization terms and maintain an optimal computational speed.
The additive learning process in XGBoost is to fit the first learner to the whole space of input data and then to fit a second model to these residuals to tackle the drawbacks of a weak learner. This fitting process will be repeated a few times until the stopping criterion is met. The ultimate prediction of the model is obtained by the sum of the prediction of each learner. To learn the set of functions used in the model, XGBoost minimizes the following regularized objective.

$$Obj = \sum_{i=1}^{N} L(y_i, \hat{y}i) + \sum_{m=1}^{M} \Omega(f_m), f_m \in F$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2$$

where L is the loss function, n is the number of observations used, Ω is the regularization term, ω is the vector of scores in the leaves, λ is the regularization parameter, and γ is the minimum loss needed to further partition the leaf node. Moreover, XGBoost can be extended to any user-defined loss function by defining a function that outputs the gradient and the Hessian (second-order gradient) and passing it through the "objective" hyper-parameter.
In addition, XGBoost implements several methods to increase the training speed of decision tree that are not directly related to the accuracy of the ensemble. In particular, XGBoost focuses on reducing the computational complexity to find the best split. This is the most time-consuming part of decision tree algorithms. Split-finding algorithms typically list all possible candidate splits and select the one with the highest gain. This requires a linear scan over each sorted attribute to find the best split for each node. To avoid repeatedly sorting the data in each node, XGBoost uses a specific compressed column-based structure in which the data is stored pre-sorted. In this way, each attribute needs to be sorted only once. This column-based storage structure allows finding the best split for each considered attribute in parallel. Instead of scanning all possible candidate splits, XGBoost implements a method based on percentiles of the data, testing only a subset of the candidate splits and calculating their gain using aggregated statistics. More detailed information and computational procedures of the XGBoost algorithm can be found in Tianqi Chen [5].

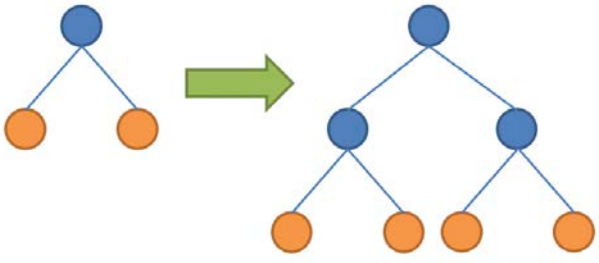


Fig. 7 – XGBoost level-wise tree growth

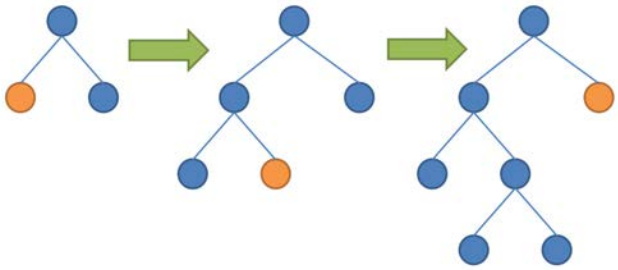


Fig. 8 – LightGBM level-wise tree growth

3.2.6 LightGBM (LGBM)

LightGBM is a gradient boosting framework that uses tree-based learning algorithms [6]. It proposed to solve the problems of Gradient Boosting Decision Tree(GBDT) in mass data. The main difference is that the decision tree in LightGBM are grown leaf-wise, as shown in Fig. 7 and Fig. 8, instead of the traditional level-wise that requires checking all of the previous leaves for each new leaf, which improves accuracy and prevents overfitting. Moreover, LightGBM uses a histogram to identify the optimal segmentation point. A histogram replaces the traditional pre-sorted, so in a sense, it sacrifices accuracy for speed. There are three aspects of differences between LightGBM and XGBoost.

- First is the computational complexity. Compared with XGBoost, LightGBM develops two kinds of methods to reduce the dimensions of input features so as to decrease the computational complexity. Based on the graph algorithm, LightGBM employs Exclusive Feature Bundling (EFB) to reduce the total number of input features. At the same time, LightGBM utilizes Gradient-based One-side Sampling (GOSS) to rank the samples according to the gradients. The proportion of features with large gradients increases by selecting the features with large gradients and combining some randomly selected features with smaller gradients.

- Second is the difference in strategy. LightGBM employs a leaf-wise strategy, while XGBoost employs a level-wise one. Resource wasting exists in XGBoost for there are indiscriminate nodes split into all layers even when the gain is minimal. On the other hand, LightGBM only splits leaf nodes with the greatest splitting gains. Such a greedy operation will also lead to overfitting and extremely large tree depth, so the tree depth in LightBGM should be constrained.

- Third is the scale of parallelization operation. Compared with XGBoost which focuses on the parallelism of features, LightGBM can parallelly deal with the features, data processing, and voting operations, which makes it be able to handle a larger data set.

3.2.7 Summary of machine learning methods

In this subsection, we give a brief summary of all the algorithms described above.

The MLP performs well when evaluating a probabilistic performance metric. However, it is not particularly better at handling high-dimensional data sets than other methods due to the large number of parameters that need to be tuned.

The main strengths of SVM are its effects on high-dimensional data and on data sets in which the number of features is greater than the number of observations. It takes less memory consumption due to the use of support vector and the use of various kernel functions, which are used in the decision function. However, SVM would overfit the model if the differences between the number of features and observations is too big.

One of the Decision Tree's major strengths is that it is easy to understand and to analyze. The disadvantage of the Decision Tree is that it is prone to overfitting and has low generalization performance.

Random Forest, XGBoost, and LightGBM all belong to ensemble methods. Ensemble methods combine the predicted results of multiple base estimators. So the results are improved as compared to some individual estimators. There are two main kinds of ensemble methods. The first one, such as Random Forest, includes techniques that consider results from many individual estimators and combine their results using the average. The second one, such as LightGBM and XGBoost, includes techniques that combine many weak estimators to get a decisive result of an ensemble.

4. EXPERIMENTATION AND EVALUATION

In this section, we use the differential data as input features and train the ML model with multiple ML algorithms. After training, we use evaluation data and different evaluation metrics to evaluate the model prediction capacity.

Note that there is no way to know in advance the best values for hyper-parameters, so ideally we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyper-parameters which improves the training efficiency.

4.1 Feature reduction

Using ensembles of decision tree methods like XGBoost may not perform well if the input features have much noise, as this can result in overfitting. In the feature vectors we use, due to the large number of features, the model training is not efficient and the training time becomes very long. Some features not only do not contribute to the modeling, but increase the complexity of the model. From the point of view of machine learning, the reason for feature reduction is to shorten the model computational time and to decrease the number of observations needed for a statistically accurate model. In terms of network management, reducing the feature set means that we can reduce the overhead of network monitoring. So trying to reduce the number of features becomes very important.

Generally, the RF and XGBoost have a built-in function that evaluates the importance of features. Some of the feature importance bar graph plots based on RF and XGBoost modeling are shown in Fig. 9. The features are sorted based on their importance.

In both RF and XGBoost, `activities/prefixes`, `sent/current-prefixes`, `prefixes/total-entries` and `as-path/total-entries` show significant importance compared to the other features. And these feature importance ranking results seem reasonable: 1) when the network goes down or the device goes down, the outgoing bytes definitely change; 2) the decline in the number of the activated links decreases the number of prefixes; 3) decrease of the prefixes changes the total number of the entries; 4) failures of the nodes absolutely affect the network-outgoing-packets.

However, there are large differences between RF and XGBoost feature importance ranking results. For example, `address-family/total-memory` has the highest rank in the result of feature ranking of the RF method, while it is not a key feature in the XGBoost method. Also, feature `network-outgoing-bytes` stands as an important role in XGBoost, while it is ranked much lower than many other features by the RF method. Some studies have reported that the feature importance ranking built-in function of RF is biased and unreliable [24]. Considering that, we choose to use the features that are ranked by XGBoost instead of ranking results by RF method.

Take the result of XGBoost ranking for consideration. If we take the peer router down as an instance, when the peer device is down, according to the default BGP configuration on the experiment routers, after 180 seconds, the link is down. Consequently, the number of next-hops is definitely changed, and also the total number of octets received in input packets from the specified address family includes those received in error. These will not only affect the incoming and outgoing bytes but also cause drops of the packets and reduction of current prefixes. Some differences in features importance ranking could be a result of features dependency. However, in general, the feature rankings obtained in this paper are reasonable and beneficial for future studies.

Dropping features with a low score (no contribution or low contribution to the model) will not influence the accuracy (Fig. 10 and Fig. 11) but benefit the model by reducing training time (Fig. 12). Fig. 10 shows the accuracy and precision using different numbers of features and Fig. 11 shows the accuracy and recall. As the number of input features changes, precision of the failures `Node Down`, `Interface Down`, `BGP Injection`, and `BGP Hijack` have maintained a relatively high accuracy rate, while `Packet Loss & Packet Delay` are relatively low. Recall of the

Fig. 9 – Feature score and importance ranking (Left: RF, Right: XGBoost)

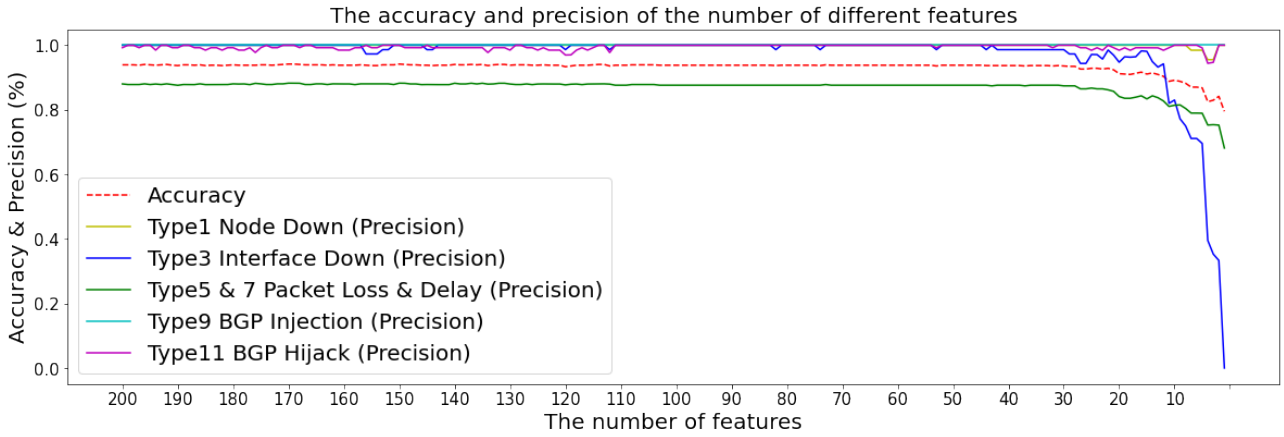


Fig. 10 – Accuracy and precision with different features

failures `Node Down`, `Interface Down`, `BGP Injection`, and `Packet Loss & Packet Delay` have maintained a relatively high accuracy rate, while `BGP Hijack` are relatively low.

The top 150 important features get the best performance which is 94.17% of average accuracy. Also, it can be seen that there are abrupt reductions on `Interface Down` and slight reductions on other failures after the number of features is reduced to 30. So we can use only 30 features to achieve a good performance, almost the same as the best performance, and for the following experimentation, we use these 30 features for training.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - measure\ score = \frac{2 * recall * precision}{recall + precision} \quad (3)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (4)$$

4.2 Evaluation

To measure ML-based classifiers' performance, we use the following evaluation metrics: *Precision*, *Recall*, *F-measure*, and *Accuracy*. These metrics are calculated by

Table 3 – Comparison of detection accuracy of different algorithms

No.	Method	Accuracy	Training Time(s)
1	XGBoost	0.9369	0.33
2	LightGBM	0.9333	2.81
3	Random Forest	0.9274	0.80
4	MLP	0.8131	1.52
5	Decision Tree	0.8095	0.38
6	SVM	0.7905	5.53

Table 4 – Test cases for different labels

TypeNo.	Type Name	Test Cases
1	Node Down	64
3	Interface Down	72
9	BGP Injection	156
11	BGP Hijack	180
57	Packet Loss and Delay	368

Eq (*(1)*), Eq (*(2)*), Eq (*(3)*), and Eq (*(4)*) assigned with the number of True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN).

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. The recall is the ratio of correctly predicted positive observations to all observations in the

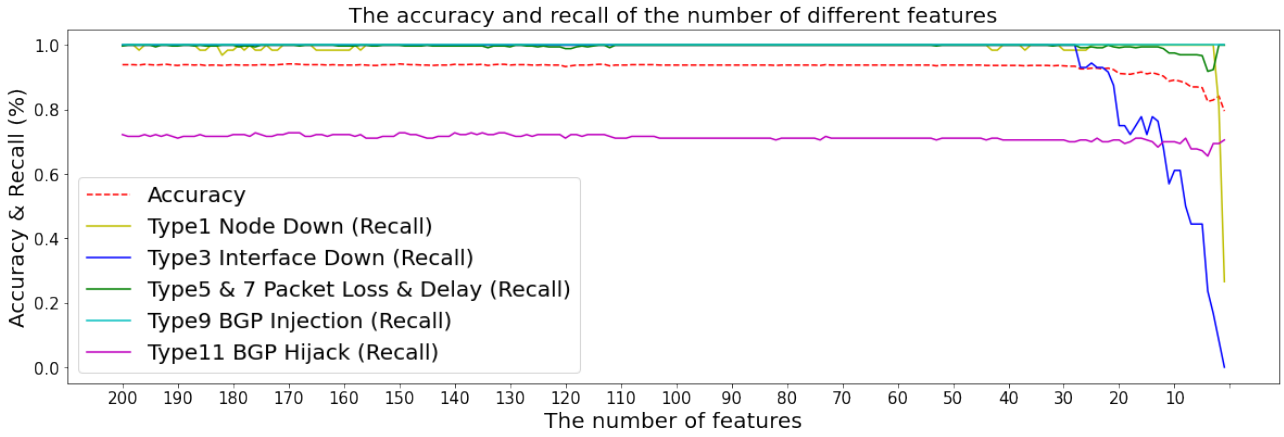


Fig. 11 – Accuracy and recall with different features

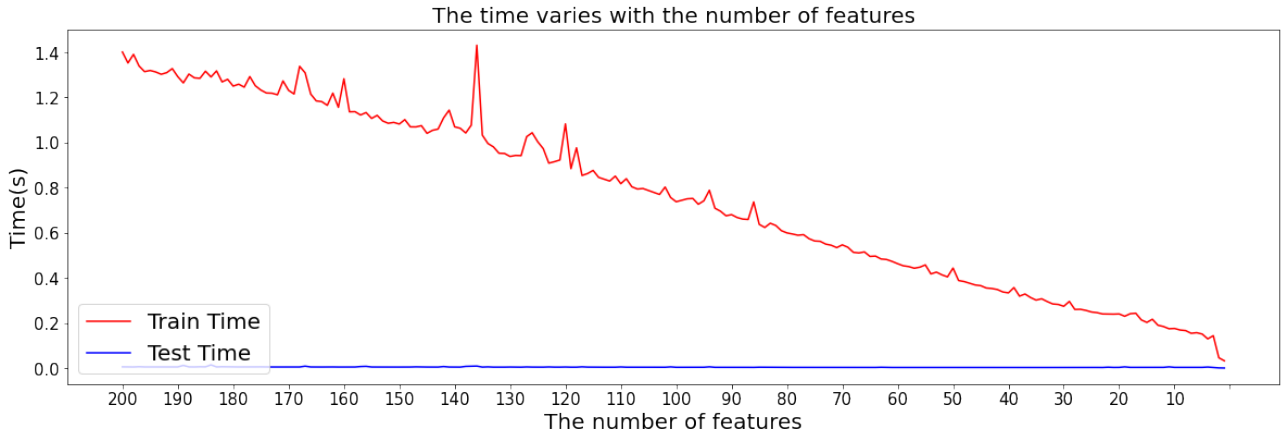


Fig. 12 – Training time with different features

Table 5 – Comparison of experimental results of different machine learning methods

Method	XGBoost			LightGBM		
Evaluation Criteria	Precision	Recall	F-measure	Precision	Recall	F-measure
1: node-down	1.00	**1.00**	1.00	1.00	**1.00**	1.00
3: interface-down	0.99	**1.00**	0.99	0.99	0.97	0.98
5, 7: tap-loss (delay)	0.88	**1.00**	0.93	0.87	**1.00**	0.93
9: ixnetwork-bgp-injection	1.00	**1.00**	1.00	1.00	**1.00**	1.00
11: ixnetwork-bgp-hijacking	1.00	0.71	0.83	1.00	0.70	0.82
Method	Random Forest			Decision Tree		
Evaluation Criteria	Precision	Recall	F-measure	Precision	Recall	F-measure
1: node-down	1.00	**1.00**	1.00	1.00	0.86	0.92
3: interface-down	0.97	**1.00**	0.99	0.82	0.89	0.85
5, 7: tap-loss (delay)	0.88	0.97	0.92	0.85	0.73	0.78
9: ixnetwork-bgp-injection	1.00	**1.00**	1.00	1.00	**1.00**	1.00
11: ixnetwork-bgp-hijacking	0.94	0.72	0.82	0.58	0.76	0.66
Method	SVM			MLP		
Evaluation Criteria	Precision	Recall	F-measure	Precision	Recall	F-measure
1: node-down	0.97	0.97	0.97	1.00	0.83	0.91
3: interface-down	0.92	0.65	0.76	0.92	0.61	0.73
5, 7: tap-loss (delay)	0.68	0.99	0.81	0.71	**1.00**	0.83
9: ixnetwork-bgp-injection	0.99	0.54	0.70	1.00	0.65	0.79
11: ixnetwork-bgp-hijacking	1.00	0.58	0.73	0.97	0.65	0.78

	1	3	9	11	57	SUM
1	64	0	0	0	0	64
3	0	72	0	0	0	72
9	0	0	156	0	0	156
11	0	0	0	128	52	180
57	0	0	0	0	368	368

(a) XGBoost

	1	3	9	11	57	SUM
1	64	0	0	0	0	64
3	0	70	0	0	2	72
9	0	0	156	0	0	156
11	0	1	0	126	53	180
57	0	0	0	0	368	368

(b) LightGBM

	1	3	9	11	57	SUM
1	64	0	0	0	0	64
3	0	72	0	0	0	72
9	0	0	156	0	0	156
11	0	0	0	130	50	180
57	0	2	0	9	357	368

(c) Random forest

	1	3	9	11	57	SUM
1	55	8	0	0	1	64
3	0	64	0	1	7	72
9	0	0	156	0	0	156
11	0	3	0	137	40	180
57	0	3	0	97	268	368

(d) Decision tree

	1	3	9	11	57	SUM
1	62	0	0	0	2	64
3	0	47	0	0	25	72
9	0	2	85	0	69	156
11	1	1	1	104	73	180
57	1	1	0	0	366	368

(e) SVM

	1	3	9	11	57	SUM
1	53	1	0	0	10	64
3	0	44	0	0	28	72
9	0	1	102	2	51	156
11	0	2	0	117	61	180
57	0	0	0	1	367	368

(f) MLP

Fig. 13 – Comparison of confusion matrix of different models

actual class. But the difference from the precision rate is that the recall rate focuses more on the proportion of samples which are True Positive (TP) that are successfully predicted. The F1 score is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account. The core idea of F1 score is to improve precision and recall as much as possible while also hoping that the differences between the two is as small as possible.

Table 3 shows the total accuracy of different models. XGBoost shows the best performance of accuracy with the least training time, and then LightGBM and Random Forest. Decision Tree, MLP, and SVM relatively show lower accuracy. Results prove the stable and outstanding performance of tree-structured models, as well as the lifting performance of the ensemble learning methods. We believe that the reason for the low accuracy of the Decision Tree is the overfitting of the training model. As for MLP, the final classification performance strongly depends on whether the optimal solution can be found. However, the back propagation algorithm in MLP tends to converge to the local optimum, so the classification accuracy cannot be ensured. SVM yields the longest training time but the lowest prediction accuracy. The reason may be concluded into the inherent computational complexity of SVM and the influence of the unrelated features in the data.

Table 5 shows the precision, recall, and F-measure results of each machine learning method on different failures. It can be seen that `Node Down` failure can be well predicted in every model with high precision and recall.

Fig. 13 shows the diagonal of the confusion matrix represents the number of samples that are correctly classified, while others are wrongly classified.

5. CONCLUSION

In this paper, we employ a highly practical and reliable approach to solve the problem about how to automatically and rapidly detect network and device failures. First, we define a staged method including feature extraction from unstructured network logs, a differential approach to highlight the differences between normal and abnormal states and several ML models to realize failure classification. Then, we apply the staged method to six popular machine learning algorithms, including Decision Tree (DT), XGBoost, LightGBM, Multilayer Perceptron (MLP), Random Forest (RF), and Support Vector Machine (SVM). After a comparative evaluation, we reveal that the tree-based models (such as XGBoost) outperform others in detecting network failures. Third, we employ a model refinement method to sort the features according to their importance score. We confirm that with the most useful features can gain computational speed without obvious degradation of accuracy. Finally, we also find that latency and loss are confused according to the RF confusion matrix so that they are hard to predict inherently.

Overall, our results achieve a reliable method for detecting network failures: almost 100% accuracy when detecting network and device failures, 86% accuracy when detecting packet loss and delay, and a total average of 94% accuracy. At the same time, the proposed feature extraction and refinement method can reduce computation without degrading the performance.

From the evaluation results of different types of models, we know that different methods are capable of learning some parts of the problem, but not the whole space of the problem. This may constitute the potential objects for future studies. We can build multiple different learners and we use them to build an intermediate prediction,

one prediction for each learned model. Then we add a new model which learns from the intermediate predictions of the same target.

6. FUTURE WORK

As for future work, we are aiming to compare different machine learning methods, for example, DT, MLP, RF, XGBoost, etc. with different data sets, not only KDDI's, to explore clearer boundaries of the most suitable range between each method so that we could take the optimal solution for diverse failures. In addition, analyzing the topology and applying graphical convolution networks at the stage of determining the importance of features might prove a promising result.

REFERENCES

[1] "Coronavirus: Five firms booming despite the lockdown," IV , {https://www.bbc.com/news/business-52383193}.

[2] "How zoom became so popular during social distancing," IV , {https://www.cnbc.com/2020/04/03/how-zoom-rose-to-the-top-during-the-coronavirus-pandemic.html}.

[3] "Google lost $1.7m in ad revenue during youtube outage, expert says," IV , {https://finance.yahoo.com/news/google-lost-1-7m-ad-144303640.html}.

[4] Junichi Kawasaki, Genichi Mouri, and Yusuke Suzuki, "Comparative analysis of network fault classification using machine learning," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.

[5] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[7] Mitchell Vásquez-Bermúdez, Jorge Hidalgo, María del Pilar Avilés-Vera, José Sánchez-Cercado, and Christian Roberto Antón-Cedeño, "Analysis of a network fault detection system to support decision making," in *International Conference on Technologies and Innovation*. Springer, 2017, pp. 72–83.

[8] Todd E Marques, "A symptom-driven expert system for isolating and correcting network faults," *IEEE Communications Magazine*, vol. 26, no. 3, pp. 6–13, 1988.

[9] Irene Katzela and Mischa Schwartz, "Schemes for fault identification in communication networks," *IEEE/ACM Transactions on networking*, vol. 3, no. 6, pp. 753–764, 1995.

[10] Anastasios T Bouloutas, George W Hart, and Mischa Schwartz, "Fault identification using a finite state machine model with unreliable partially observed data sequences," *IEEE Transactions on Communications*, vol. 41, no. 7, pp. 1074–1083, 1993.

[11] Cynthia S Hood and Chuanyi Ji, "Proactive network-fault detection [telecommunications]," *IEEE Transactions on reliability*, vol. 46, no. 3, pp. 333–341, 1997.

[12] Eleftheria Athanasopoulou and Christoforos N Hadjicostis, "Probabilistic approaches to fault detection in networked discrete event systems," *IEEE transactions on neural networks*, vol. 16, no. 5, pp. 1042–1052, 2005.

[13] İlhan Firat Kilinçer, Fatih Ertam, Orhan Yaman, and Ayhan Akbal, "Automatic fault detection with bayes method in university campus network," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*. IEEE, 2017, pp. 1–4.

[14] Marc Ruiz, Francesco Fresi, Alba P Vela, Gianluca Meloni, Nicola Sambo, Filippo Cugini, Luca Poti, Luis Velasco, and Piero Castoldi, "Service-triggered failure identification/localization through monitoring of multiple parameters," in *ECOC 2016; 42nd European Conference on Optical Communication*. VDE, 2016, pp. 1–3.

[15] Carla Sauvanaud, Kahina Lazri, Mohamed Kaâniche, and Karama Kanoun, "Anomaly detection and root cause localization in virtual network functions," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 196–206.

[16] Srinikethan Madapuzi Srinivasan, Tram Truong-Huu, and Mohan Gurusamy, "Machine learning-based link fault identification and localization in complex networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6556–6566, 2019.

[17] Rolf Stadler, Rafael Pasquini, and Viktoria Fodor, "Learning from network device statistics," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 672–698, 2017.

[18] Karwan Qader, Mo Adda, and Mouhammd Al-Kasassbeh, "Comparative analysis of clustering techniques in network traffic faults classification," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 4, pp. 6551–6563, 2017.

[19] Vladimir N Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[20] Alex J Smola and Bernhard Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[21] Nello Cristianini, John Shawe-Taylor, et al., *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.

[22] Stanislaw Osowski, Krzysztof Siwek, and Tomasz Markiewicz, "Mlp and svm networks-a comparative study," in *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004.* IEEE, 2004, pp. 37–40.

[23] Tin Kam Ho et al., "Proceedings of 3rd international conference on document analysis and recognition," in *IEEE*, 1995, pp. 278–282.

[24] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–21, 2007.

AUTHORS

Xia Fei is currently a master student at the Graduate School of Interdisciplinary Information Studies (GSII), the University of Tokyo. His research work focuses on practical approaches to the AI system. He is interested in doing research in network fault detection, data analysis, and machine learning.

Aerman Tuerxun received his B.Eng of Information Security from the University of Science and Technology of China and is currently pursuing his master degree at the Graduate School of Interdisciplinary Information Studies (GSII), the University of Tokyo. His research interests include local 5G, network slicing, network intelligence etc.

Lu Jiaxing is a Ph.D student at the Graduate School of Interdisciplinary Information Studies (GSII), the University of Tokyo. His research interests include federated learning and edge computing.

Ping Du received a Ph.D. from the Graduate University for Advanced Studies in Japan in 2007. From 2008, he worked for the National Institute of Information and Communication Technologies (NICT) of Japan. Now, he works for the University of Tokyo as a project associate professor. His research interests include optical networks, network security, network virtualization, mobile networks and machine learning etc.

Akihiro Nakao received his BS in physics and ME in information engineering from the University of Tokyo. He worked at IBM Yamato Laboratory, Tokyo Research Laboratory, and IBM Texas Austin. He received his MS and PhD in computer science from Princeton University. Since 2005, he has been an associate professor and then a professor in applied computer science at the Interfaculty Initiative in Information (III) Studies, Graduate School of Interdisciplinary Information Studies, the University of Tokyo. Now, he is a professor in Faculty of Engineering, University of Tokyo. He has been appointed Chairman of the 5G Mobile Network Promotion Forum (5GMF) Network Architecture Committee by the Japanese government.

SIMULATION OF MACHINE LEARNING-BASED 6G SYSTEMS IN VIRTUAL WORLDS

Ailton Oliveira[1], Felipe Bastos[1], Isabela Trindade[1], Walter Frazão[1], Arthur Nascimento[1], Diego Gomes[2], Francisco Müller[1] and Aldebaro Klautau[1]
[1]Universidade Federal do Pará - LASSE — www.lasse.ufpa.br, Av. Perimetral S/N, Belém, Pará, Brazil., [2]Universidade Federal do Sul e Sudeste do Pará - IGE — www.ige.unifesspa.edu.br , Marabá, Pará, Brazil.

NOTE: Corresponding author: Ailton Oliveira, ailton.pinto@itec.ufpa.br

Abstract – *Digital representations of the real world are being used in many applications, such as augmented reality. 6G systems will not only support use cases that rely on virtual worlds but also benefit from their rich contextual information to improve performance and reduce communication overhead. This paper focuses on the simulation of 6G systems that rely on a 3D representation of the environment, as captured by cameras and other sensors. We present new strategies for obtaining paired MIMO channels and multimodal data. We also discuss trade-offs between speed and accuracy when generating channels via ray tracing. We finally provide beam selection simulation results to assess the proposed methodology.*

Keywords – 6G, artificial intelligence, machine learning, MIMO, ray tracing

1. INTRODUCTION

Machine Learning (ML) and, more generally, Artificial Intelligence (AI), are currently under investigation to optimize the performance of future communication networks [1]. The applications include, for instance: physical layer (PHY) optimizations, network management and self-organization [2, 3]. Given the increasing importance of ML/AI in communications, there are several initiatives concerning ML/AI architectures, such as the one carried out by ITU [4]. This trend should continue with 6G systems, which are expected to support augmented reality, multisensory communications and high-fidelity holograms [5]. One such application is autonomous driving, where digital representations are used to generate sensors for hardware-in-the-loop testing[1]. And because such digital representations of the world will flow through the 6G network, it is expected that ML/AI can leverage them. Therefore, a specific set of simulation tools for future networks is characterized by the requirement of being able not only of dealing with communication channels, but also the corresponding sensor data, matched to the *scene*.

This paper focuses on strategies for simulating 6G systems that require a representation of the environment, as captured by cameras and, eventually, additional modalities of sensors. More specifically, we consider Multiple Input / Multiple Output (MIMO) systems and discuss the required generation of channels that are consistent with the scene at each time instant. A simulation that integrates communication networks and artificial intelligence immersed in virtual or augmented reality can be computationally expensive, especially for time-varying digital worlds. We discuss two categories of simulations: the one in which the ML/AI model is executed within the virtual world simulation loop and the one in which the ML/AI model is out of the loop and the simulator can then write files to be later used for training ML/AI models. An example of the first category (INLOOP is going to be used as the *UFPA Problem Statement* [6] for the *2021 ITU AI/ML in 5G Challenge*.

Concerning the channel generation, the requirement of having an associated digital world precludes the adoption of a class of modern channel models that are not related to any virtual world representation, such as the ones presented in [7, 8]. We therefore adopt ray tracing (RT for MIMO channel generation, which is aligned with other recent work (see, e. g. [1] and references therein and allows the generation of site-specific communication channel responses with temporal and spatial consistency.

Another motivation for this paper is to promote public datasets. In many ML application domains, the data is abundant or has a relatively low cost. For example, the deep learning-based text-to-speech system presented in [9], which represents the state-of-the-art, achieves quality close to natural human speech after being trained with 24.6 hours of digitized speech. In contrast, the research and development of 5G has to deal with a relatively limited amount of data. Considering the 5G research on AI/ML applied to millimeter waves (mmWave MIMO, the lack of abundant data from measurements or simulations hinders some data-driven lines of investigation. With 6G moving towards the use of even higher (Terahertz frequency bands [10], it becomes even more challenging to perform measurement campaigns for this frequency range [11], particularly for outdoor environments. Given that channel measurements for 6G will demand relatively expensive equipment, the simulation strategies for

[1]https://www.ni.com/pt-br/innovations/white-papers/17/altran-and-ni-demonstrate-adas-hil-with-sensor-fusion.html

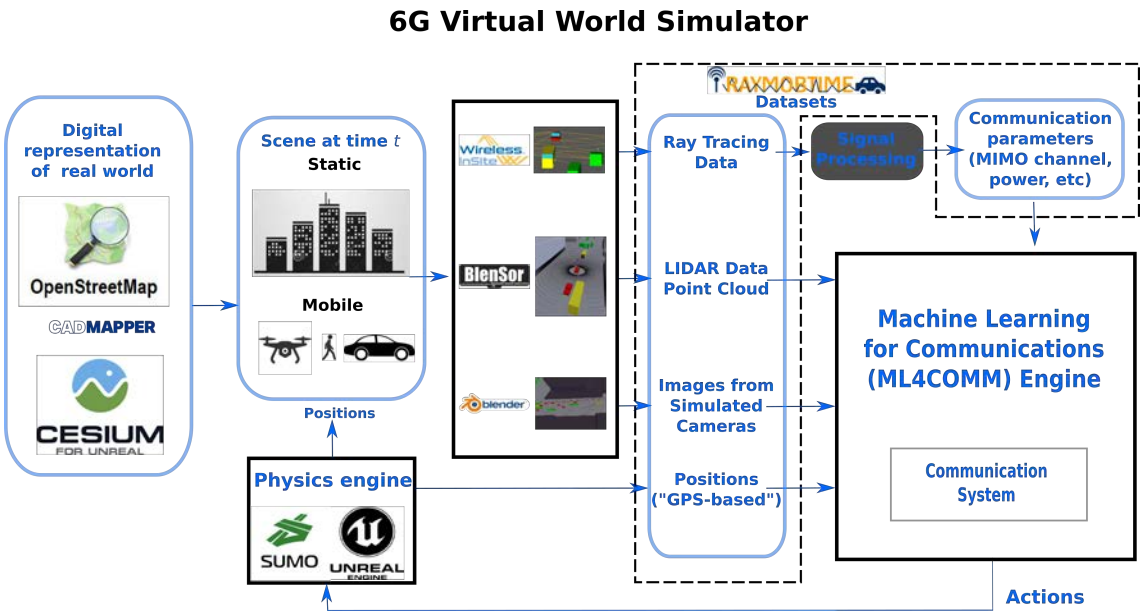


Fig. 1 – Block diagram of CAVIAR simulation with AI/ML in the simulation loop (INLOOP). In OUTLOOP simulations, the simulator can write files that will be later used for designing and assessing AI/ML models.

modeling mobility and virtual worlds discussed in this paper can alleviate the problem. The generated datasets are especially useful when spatial consistency and time evolution are important to assess an AI/ML technique applied to the physical layer.

The contributions of this paper are:

- A discussion of strategies and software for simulating *Communication networks and Artificial intelligence immersed in VIrtual or Augmented Reality* (CAVIAR).
- A preview of a CAVIAR simulator that will be used in the *UFPA Problem Statement* for the *2021 ITU AI/ML in 5G Challenge*, which consists of a Reinforcement Learning (RL) problem with the decisions taken by the RL agent changing the virtual world on-the-fly (as the simulation evolves).
- Discuss a new methodology using photogrametric data available from the Internet to improve the realism of ray-tracing simulations by automatically assigning electromagnetic properties to the materials composing a scene, via semantic segmentation with deep neural networks.
- Results exposing trade-offs between speed and accuracy when generating channels via ray tracing.
- Results of a reinforcement learning experiment in beam selection realized in the CAVIAR environment.
- Source code and datasets to reproduce the baseline of *2021 ITU AI/ML in 5G Challenge*.[2]

[2] https://ai5gchallenge.ufpa.br/

The rest of the paper is organized as follows. Methods and software for CAVIAR simulation of 6G are presented in Section 2. Section 3 explains some improvements in the RT simulation methodology. Section 4 presents numerical results and their discussion. Finally, Section 5 concludes the paper.

2. 6G SIMULATION IN VIRTUAL WORLDS

Gaming and other industries are driving the development of sophisticated tools to create virtual worlds, composed of 3D models, physics engines and other components. The virtual world 3D scenery can be created from scratch by 3D design modelers, or from data imported from the real world. For instance, the new *Cesium* plug-in for *Epic Game*'s *Unreal Engine*[3] integrates photogrametric information obtained from drones into 3D models available via *Cadmapper*[4] and other sites. This complements tools such as *Twinmotion*,[5] which facilitate the construction of 3D virtual worlds. This paper promotes the vision that 6G and beyond will benefit from the availability of virtual worlds to leverage ML/AI applied to communication networks. Current investigations of AI applied to 5G aim at finding how raw data from sensors such as LIDAR and cameras can optimize the communication performance [12, 13, 14, 15]. But the possibility of having realistic 3D models, physics engines and other virtual reality assets for simulations of communication systems, opens new horizons in terms of AI/ML applied to 6G and beyond.

[3] https://cesium.com/blog/2021/03/30/cesium-for-unreal-now-available/.
[4] https://cadmapper.com.
[5] https://www.unrealengine.com/en-US/twinmotion.

As proposed in [16], the CAVIAR framework concerns a specific category of 6G simulations that rely on virtual worlds and incorporate two subsystems: wireless communications and AI/ML. In the next paragraphs, we briefly review the CAVIAR framework, depicted in Fig. 1, and then focus on the important aspect of generating the communication channel corresponding to a given scene of the virtual world. We discuss how the Raymobtime methodology [12] fits well to the demand for communication channels imposed by 6G CAVIAR simulations.

A CAVIAR simulation generates multimodal data for each discrete time $t \in \mathbb{Z}$, and is able to operate in two modes, the first mode is focused on online learning, running the simulation and the neural network simultaneously, creating an environment where data is transmitted in real time, or in discrete samples with time stamps defined by the user. The second mode of operation performs data recording in databases or text files, working as a tool for creating datasets. Along the simulation, the *machine learning for communications* (ML4COMM) engine operates on data organized as an *episode* $E = [(\mathcal{P}_1, \mathcal{O}_1), \ldots, (\mathcal{P}_S, \mathcal{O}_S)]$, with a sequence of S tuples $(\mathcal{P}_t, \mathcal{O}_t), t = 1, \ldots, S$, of *paired* data, where $\mathcal{P}_t$ and $\mathcal{O}_t$ are sets with the input AI/ML parameters and corresponding outputs, respectively. In supervised learning, $\mathcal{O}_t$ consists of desired labels for classification or regression, while for reinforcement learning $\mathcal{O}_t$ consists of rewards for the agents. The tuples $(\mathcal{P}_t, \mathcal{O}_t)$ denote evolution over discrete-time t. In our methodology, the outputs of the simulators are periodically stored as "snapshots" (or *scenes*) over time tT_{sam}, where T_{sam} is the sampling period and $t \in \mathbb{Z}$.

The main steps in Fig. 1 can be summarized as follows. The environment is composed of a 3D scenery with fixed and mobile objects. These objects are created and placed with specialized tools and data from the Internet, as described in [12] and [17]. The positions and interactions among mobile objects are determined by a *physics engine* (for instance, the Unreal engine or the Simulation of Urban MObility (SUMO) traffic generator [18]).

Once the *scene* is complete, the environment is represented via sensors, such as LIDAR, which is simulated by Blensor and Blender software, returning point cloud data (PCD) that maps the shapes of the 3D space around the sensor. It is possible to adjust the resolution of the PCD through a quantization process. A ray-tracing software (Remcom's Wireless InSite in Fig. 1) also captures the communication channel for the given scene. The sensors output constitute the episode input $\mathcal{P}_t$, and the corresponding output $\mathcal{O}_t$ is obtained by a *signal processing* module. These episodes are actually what is stored in Raymobtime episodes [12] but in a CAVIAR simulation they can be created and used on-the-fly, if needed. The CAVIAR 6G virtual world simulator also incorporates a communication system that has some functionalities driven by the ML4COMM engine. The ML4COMM engine also relies on the scene description and can extract features from the raw sensor data to feed its AI/ML algorithms.

Fig. 1 illustrates the INLOOP CAVIAR framework with the AI/ML module within the simulation loop. When the decisions of this module do not affect the environment, it can be convenient to split the simulation into two stages, with the first one being an OUTLOOP CAVIAR simulation that writes *episode* files that will be later used for designing and assessing AI/ML models. The more evolved INLOOP simulation is required in cases such as a drone mission in which the AI/ML decisions will change the drone trajectory and, consequently, its wireless channel. In general, when the AI/ML model issues commands or actuator signals that effectively change the trajectories of mobile entities, alter the environment or the communication system state (e.g., buffer occupation), the simulations may need to be INLOOP and communication channels generated on-the-fly. In the simpler OUTLOOP simulation category, channels can be pre-computed and the communication simulation decoupled from the physical engine, as often used in AI/ML applied to beam selection [19, 12]. The next sections provide two examples to distinguish INLOOP and OUTLOOP CAVIAR simulations.

2.1 OUTLOOP CAVIAR simulation for beam selection

Beam selection is a classical application of AI/ML to communications [20, 21, 22]. The goal is to choose the best pair of beams for analog beamforming, with both transmitter (Tx) and receiver (Rx) having antenna arrays with only one Radio Frequency (RF) chain and fixed beam codebooks. Fig. 2 illustrates beamforming from a Base Station (BS) to both vehicles and drones.

Fig. 2 – Beamforming from BS to both vehicles and drones.

We first assume beam selection for a vehicular to infrastructure network, to illustrate an OUTLOOP CAVIAR simulation. In this case the communication subsystem is a downstream MIMO system in which a BS with a Uniform Linear Array (ULA) of N_t antennas communicates with cars with ULAs of N_r antennas. ML is used for beam-selection.

Discrete Fourier Transform (DFT) codebooks $\mathcal{C}_t = \{\bar{\mathbf{w}}_1, \cdots, \bar{\mathbf{w}}_{N_t}\}$ and $\mathcal{C}_r = \{\bar{\mathbf{f}}_1, \cdots, \bar{\mathbf{f}}_N\}$ are used at the transmitter and the receiver sides, respectively. The beam pair $[p, q]$ is converted into a unique index $i \in \{1, 2, \cdots, M\}$, where $M \leq N_t N_r$. For the i-th pair, the *equivalent channel* (without considering noise) can be calculated as

$$y_i = \mathbf{w}_i^* \mathbf{H} \mathbf{f}_i, \qquad (1)$$

and the *optimal* beam pair index $\hat{i}$ is given by

$$\hat{i} = \arg \max_{i \in \{1, \cdots, M\}} |y_i|. \qquad (2)$$

The beam selection is then posed as a top-k classification problem. At time t, the classifier inputs are features obtained from $\mathcal{P}_t$ and the output is the beam pair i.

For the scenario presented in this section, the trajectory of vehicles and all mobile objects do not depend on the AI/ML model, hence all the episodes can be pre-computed. Next, we discuss a simulation in which the trajectories are determined by the AI/ML model and the channels cannot be pre-computed.

2.2 INLOOP CAVIAR simulation with drones and reinforcement learning

Unmanned Aerial Vehicles (UAVs) are being used in many connected applications, such as surveillance and product delivery. UAVs can also be used as mobile radio base stations to extend reach or improve network capacity, mainly in situations of disasters and accidents. In order to meet the requirements of all these use cases, the network links need to obey particular requirements, ranging from very low latency to high data rates [23]. All this motivates intense research on 5G technologies for supporting UAV-based applications. However, there are currently few simulation tools for testing and studying telecommunication systems that involve UAV solutions and their corresponding channels. The CAVIAR framework is deeply integrated with the Unreal Engine development kit and the *Airsim* simulator [24], which bring realism to the physical aspects of the simulations.

As part of the *UFPA Problem Statement* for the *2021 ITU AI/ML in 5G Challenge*, we designed an INLOOP CAVIAR simulation in which RL is executed at the BS and used in two problems: a) determine the drone trajectory and b) beam selection along the downstream. In the challenge, the drones need to deliver pizzas to distinct addresses in a neighborhood. Fig 3 illustrates the scenario.

Fig. 3 – Scene from an INLOOP CAVIAR simulation in which a drone is served by a BS and RL is used for beam selection and for determining the drone trajectory.

The scenario depicted in Fig 3 allows us to investigate several problems that relate communication with drones path planning. One important issue is how to obtain the channels on-the-fly. If the visualization is performed after the whole simulation is finished, the time to generate the channel (via RT, for instance) may be longer. But in this case the scenes need to be visualized along the simulations (as part of a game, for example), then the minimum number of frames per second will impose a limit on the time to generate the communication channels.

The next section discusses our Raymobtime methodology and the corresponding datasets. Other publicly available RT-based datasets are listed in Table 1. The ViWi dataset, presented in [13], provides similar output data compared to Raymobtime, including visual data. The DeepMIMO dataset [25] is maintained by the same group as ViWi and offers only wireless channel information. The dataset described in [1] does not have visual information as well. One of the main differences between these three datasets and Raymobtime is how mobility is handled. The Raymobtime methodology simulates realistic traffic with several moving vehicles using the SUMO software in order to provide better spatial and temporal consistency, as well as channel variability due to the moving scatterers. ViWi [13] (in its first version), DeepMIMO and the map-based channel model in [1] use a fixed grid for Tx-Rx positions and therefore does not consider varying speeds for moving transceivers. ViWi version 2 provides one new scenario that includes several moving vehicles, each equipped with an omnidirectional antenna.

3. IMPROVEMENTS ON RAYMOBTIME METHODOLOGY

The Raymobtime methodology proposed in [12] aims at providing a multimodal dataset, including RT channel information and data from sensors, such as images, LIDAR and location, as illustrated in Fig. 1. One major challenge in building the Raymobtime datasets is to provide accurate wireless communication channel parameter through the use of RT simulation software. In this work, Remcom's Wireless InSite (WI) RT software [26] was adopted given

Table 1 – Other publicly available RT datasets.

Dataset name	Data Types	Environment	Frequency (GHz)	File format
ViWi [13]	Image, depth-map, wireless channel, and user location	Outdoor	28 and 60	Matlab, JPEG
DeepMIMO [25]	Wireless channel parameters	Indoor and Outdoor	2.5, 3.5, 28, and 60	Matlab
Map-based channel model [1]	Wireless channel parameters	Indoor and Outdoor	28	Matlab

its widespread use [1]. This section summarizes two improvements toward more realistic datasets for AI/ML involving MIMO channels. More details can be found in [16].

The first improvement compared to previous versions of the Raymobtime methodology is the correction of the orientation of the antenna arrays mounted on moving vehicles, so that the array follows the direction of the vehicle. As mobile objects (vehicles, people, etc.) move in the virtual world, previous versions of Raymobtime datasets were not updating the orientation of the antenna array.

The other improvement is the simulation of antenna arrays inside the RT software. Previous versions of Raymobtime always considered omnidirectional antennas inside the RT simulation. This procedure is called here Single Input, Single Output RT (SISO-RT). MIMO channel matrices are obtained during post-processing with the use of the geometrical channel model [27]. This approach reduces processing time and make the dataset more flexible, as the user can define the desired antenna arrays for all transceivers during post-processing, without the need to run RT simulations for every antenna array configuration. However, the geometrical channel model assumes planar-wave propagation, which can be problematic when using large antenna arrays [1]. A more realistic, albeit computationally expensive, alternative is to simulate the antenna arrays inside the RT processing, called MIMO-RT procedure in [16]. Each ray has its own time of arrival and angle offsets, which is equivalent to the spherical-wave assumption [1]. As shown in [28], the difference in estimated MIMO channel capacity can be quite large between the two approaches.

Table 2 presents a list of current Raymobtime datasets and their features. The datasets s011 and s012 include the improvements described in this section. The Raymobtime datasets are divided in several episodes, each one composed by a number of scenes. The smaller the time between scenes, the more similar are consecutive scenes within an episode and, consequently, the more correlated are the communication channels of a given receiver along with the scenes. Currently, RT simulations using Remcom's Wireless InSite (WI) RT software [26] are limited to sub-THz frequencies (up to 100 GHz). More details about the methodology can be found in [12].

The RT simulations demand the identification of the material of the surfaces, in order to properly simulate the electromagnetic interaction of the waves with the objects. The disposition and diversity of these materials directly impact the quality of the channels [29], making this assignment manually a time-consuming and laborious process, and usually results in few materials being actually adopted. To optimize this procedure, the next paragraphs describe ongoing research to develop a methodology to automatically assign such materials to 3D objects via semantic segmentation with deep neural networks.

Fig. 4 – Analysis region image taken from Cesium database.

Fig. 5 – Segmented version using PyTorch of the Cesium image.

Semantic segmentation is a modern approach that performs classification at pixel level, and allows us to determine both the class of an object and the boundaries of each object [30]. Current approaches of this method use deep learning in order to overcome traditional object segmentation, allowing us to classify pixels not only by their colors, but also considering the region context [31]. Due to the fact that the 3D environment is built reproducing real locations, it is possible to use databases such as Cesium and Google's Street View to obtain detailed image data from the analysis region. We are applying semantic segmentation in images obtained via the Cesium plug-in

Table 2 – Some Raymobtime datasets.

Dataset name	Frequency (GHz)	Number of receivers and type	Time between scenes (ms)	Time between episodes (s)	Number of episodes	Number of scenes per episode	Number of valid channels
s001	60	10 Mobile	100	30	116	50	41 K
s005	2.8 and 5	10 Fixed	10	35	125	80	100 K
s006	28 and 60	10 Fixed	1	35	200	10	20 K
s008	60	10 Mobile	-	30	2086	1	11 K
s011 (new)	60	10 Mobile	500	6	76	20	13K
s012 (new)	60	10 Fixed	500	6	105	20	21K

Fig. 6 – Analysis region image from Google's Street View.

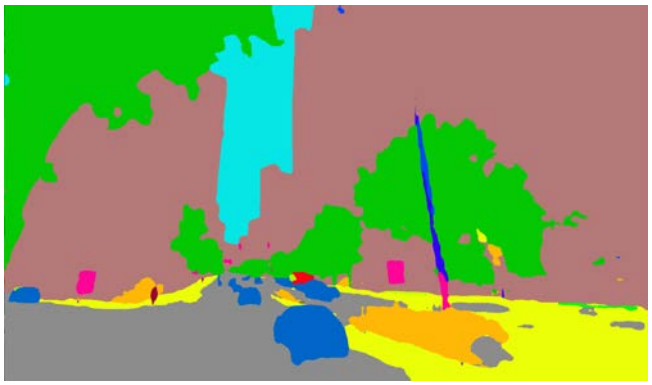


Fig. 7 – Segmented version of the Google's Street View image.

for Unreal in order to identify the different surface types which composes the scenario.

Fig. 4 and Fig. 5 show an image taken from Cesium and its segmentation, respectively. This segmentation used a PyTorch implementation of semantic segmentation models on the MIT ADE20K [32] scene parsing dataset. In this example, it is possible to verify that the algorithm was capable of determining the contour of the asphalt. On the other hand, the regions corresponding to buildings, cars and vegetation were associated to the same class. This is due to the bad quality of the images taken from Cesium, where some regions of the figure were rendered with deformations and inadequate color assignment to objects, as observed in the tree at the bottom right corner and the objects at the sidewalks, for instance. This is a challenging case for semantic segmentation. In Fig. 6 and Fig. 7, it is possible to verify that there is a significant improvement in the segmentation performance (Fig. 7) compared to the previous example (Fig. 5) when using images obtained from Google's Street View due to the better quality of the source image (Fig. 6). The segmentation was able to identify cars, asphalt, sidewalks, vegetation and buildings with a much better resolution, allowing us to classify the materials with more diversity. Our research efforts are now dedicated to mapping the stitched 2D images to the 3D model and include semantic segmentation results into RT simulations.

4. CAVIAR SIMULATION RESULTS

In this section, we discuss some key issues related to CAVIAR simulations. We start by evaluating the computational cost of RT. A snapshot of dataset s012 was simulated with different parameters, assuming isotropic antennas for SISO-RT simulations, and Uniform Linear Array (ULA) for MIMO-RT simulations. The simulations include one transmitter and 10 receivers, each with its own antenna or antenna array, depending on the scenario. The aim is to analyze the impact of the ray spacing, the use of Diffuse Scattering (DS) and the number of antenna elements in the ULA (for MIMO-RT) on the RT simulation time. DS is enabled in all SISO-RT simulations where the carrier frequency is above 6GHz (except for the datasets s011 and s012, as they were designed for the comparison between SISO-RT and MIMO-RT results. The later one has an exponential increase in simulation time when running with DS). For all the simulation results presented here, a PC with an NVIDIA RTX 2070 was used.

In the RT simulations, the transmitter shoots rays in a sphere through the scenario to find viable paths between transmitter and receiver. The minimum angle between the rays is defined as the *ray spacing*. The values in Table 3 show that the ray spacing has a great impact in the total simulation time. For SISO-RT, a simulation using a ray spacing of 0.1° takes 11 times longer than the one with ray spacing equal to 1°. For MIMO-RT, the simulation considering 0.1° ray spacing is 6.2 times longer compared to ray spacing of 1°. For context, Wireless InSite recommends setting ray spacing to 0.2° or less, for 500 m $\times$ 500 m areas [33].

DS is a special type of ray interaction with surfaces, allowing for the simulation of scattered paths caused by irregularities in materials. It increases the number of simulated paths and, consequently, the number of calculations and the run time. Table 3 presents results for simulations

with DS enabled, both in SISO-RT and MIMO-RT scenarios, considering a ray spacing of 0.5°. For SISO-RT, the run time was 87 longer when enabling the DS compared to not using it. For MIMO-RT this value was even greater: DS increased the simulation time more than 600 times.

As described in Table 4, the simulation times depend on the number of antenna elements in each Tx-Rx pair. Increasing the number of antenna elements in each Tx-Rx pair significantly raises the simulation time. A twelve-fold increase occurs when using $N_t = 64$ and $N_r = 64$ - where N_t and N_r are the number of antenna elements in the ULA of the transmitter and receiver, respectively - compared to the baseline case where $N_t = 64$ and $N_r = 2$.

Table 3 – Simulation time increase factor for one RT simulation (s012) for different ray spacing values, with and without diffuse scattering enabled. The baseline time for SISO-RT is 00:00:11.749 and for MIMO-RT (with $N_t = 64$ and $N_r = 8$) is 00:00:39.654. The time format is (HH:MM:SS.ccc).

	Simulation time increase factor	
Ray Spacing (°)	SISO-RT	MIMO-RT
1	1	0.7
0.5	1	1
0.25	2.4	1.5
0.1	11	4
0.5 (DS-enabled)	84.7	412.9

Table 4 – Simulation time increase factor for one RT simulation (s012) considering different numbers of antenna elements in the transmitter and receiver antenna arrays. The baseline time is 00:00:18.437 (with $N_t = 64$ and $N_r = 2$). The time format is (HH:MM:SS.ccc).

N_t	N_r	Simulation time increase factor
64	2	1
64	8	2.2
64	64	12 Γ

As an illustration of an INLOOP CAVIAR simulation, we developed code for the Unreal Engine and AirSim to simulate a BS serving a UAV. There are two RL agents: one for determining the UAV trajectory and the other for beam selection. We discuss only the latter agent in this paper. As the UAV flies along its trajectory, the MIMO channel is obtained according to the well-known *geometric model*, with parameters for three multipath components obtained from probability distributions (see, e. g. [27, 16]). This simpler methodology was adopted to speed up the simulations and allow for visualizing the UAV as it flies. In this specific scenario, RT channel responses are not used due to the required simulation time. The BS used a ULA with $N_t = 64$ antennas, while the UAV uses a single antenna. A DFT codebook is adopted.

At each time t, the UAV informs its position to the BS, which can then calculate the Angle of Arrival (AoA) θ of the beam at the UAV. This angle is used as the input for two beam selection algorithms: one based in RL and a simple baseline. To perform beam-selection using RL, we used a Deep Q Network (DQN) [34]. The Stable Baseline API[6] with default DQN parameters was adopted. The reward is the magnitude of the equivalent channel, as defined in Eq. *(1)*. The baseline algorithm adopts the following heuristic: it simply chooses the beam that points to the straight path direction between the BS and the UAV. For most of the UAV's path, there is Line-Of-Sight (LOS) and this heuristic achieves good results. As expected, this strategy does not work well when the link is Non-LOS (NLOS), which occurs for the angular range $\theta \in [20, 30]$ degrees.

The results of this simple experiment is provided in Fig. 8. The bottom plot shows the angle θ as the UAV takes off ($t \in [0, 25000]$), reaches its destiny and lands ($t > 76000$). During three time intervals (including a very short one) the link between the UAV and BS was NLOS. The top plot shows the magnitude of the equivalent channel $|y_{i,t}|$, in which the i-th codebook index was chosen at time t. The optimum value, obtained by exhaustively trying all $N_t = 64$ indices, is shown together with the values obtained by the DQN (RL) and baseline. While the optimum value is always larger than 5 and has an average value of 6.81, both baseline and RL struggle to reach good results and achieve average values $\mathbf{E}[|y_{i,t}|] = 1.7$ and 2.3, respectively. It should be noticed that in this case the RL agent should choose one among 64 indices having a single input (angle θ). In the *UFPA Problem Statement* for the *2021 ITU AI/ML in 5G Challenge* [6], a richer set of input features will be adopted, allowing not only beam selection but also UAV path planning.

5. CONCLUSIONS

This paper presented strategies and software for simulating 6G systems that represent the surrounding environment with images and other types of data. The so-called CAVIAR framework benefits from virtual reality tools, emphasizing the physical aspects of the movement of objects. This visual information, coupled with MIMO channels generated through RT methods, enables investigating new AI/ML algorithms in 6G that rely on the environment and learning from experience.

We also discussed how semantic segmentation and sensible RT parameters can improve generated MIMO channels. We advocate that aiming at realistic simulations is the natural path to gain a better understanding on how ML/AI can make communication systems more efficient. The effort along the direction of larger and realistic datasets is important for properly evaluating ML-based algorithms, and to avoid unfair comparisons to conventional signal processing.

[6] https://stable-baselines.readthedocs.io/en/master.

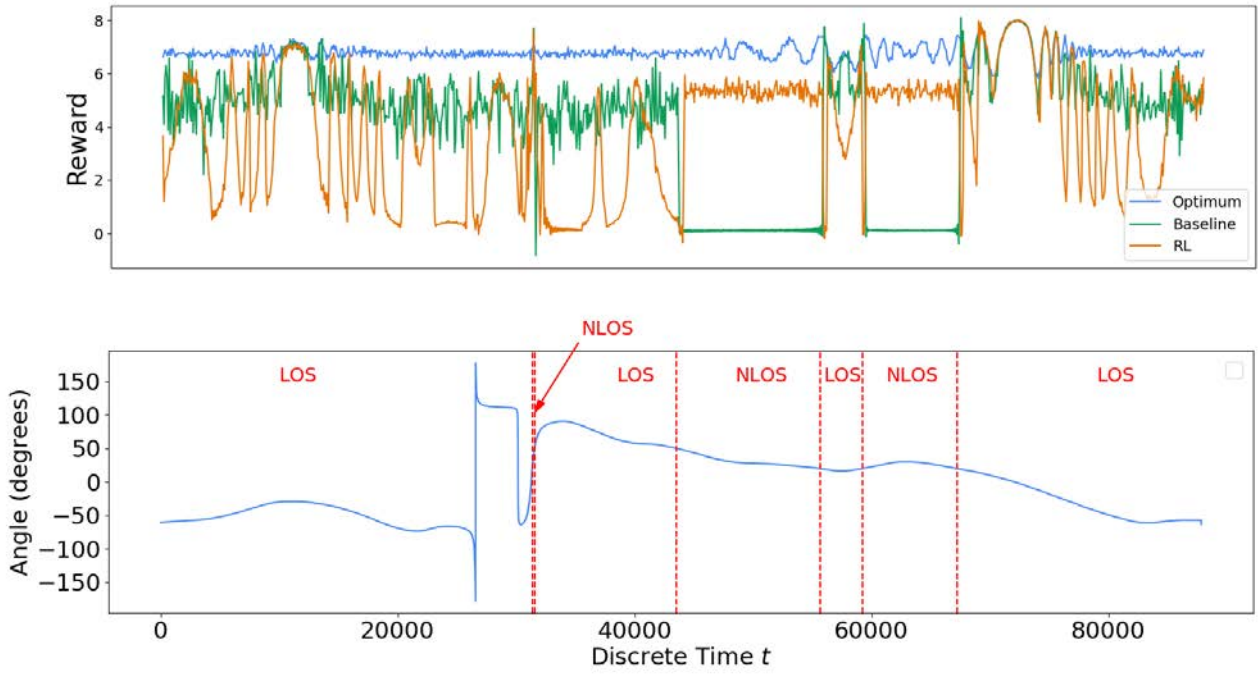


Fig. 8 – Beam selection results for a BS serving a UAV comparing RL versus a simple baseline algorithm. The optimal result (best beam pair) is also included. The top plot presents the reward is the magnitude of the equivalent channel for the i-th beam pair at the time t (higher reward values are better). The bottom plot shows the AoA θ at the UAV at each time t.

REFERENCES

[1] Y.-G. Lim, Y. J. Cho, M. S. Sim, Y. Kim, C.-B. Chae, and R. A. Valenzuela. "Map-Based Millimeter-Wave Channel Models: An Overview, Data for B5G Evaluation and Machine Learning". In: *IEEE Wireless Communications* 27.4 (Aug. 2020). Conference Name: IEEE Wireless Communications, pp. 54–62. ISSN: 1558-0687. DOI: 10.1109/MWC.001.1900315.

[2] Hongji Huang, Song Guo, Guan Gui, Zhen Yang, Jianhua Zhang, Hikmet Sari, and Fumiyuki Adachi. "Deep Learning for Physical-Layer 5G Wireless Techniques: Opportunities, Challenges and Solutions". In: *IEEE Wireless Communications* 27.1 (2020), pp. 214–222. DOI: 10.1109/MWC.2019.1900027.

[3] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. "Deep Learning in Mobile and Wireless Networking: A Survey". In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2224–2287. DOI: 10.1109/COMST.2019.2904897.

[4] *Focus Group on Machine Learning for Future Networks including 5G*. https://www.itu.int/en/ITU-T/focusgroups/ml5g/Pages/default.aspx. (Accessed on 03/19/2020).

[5] C. De Lima et al. "Convergent Communication, Sensing and Localization in 6G Systems: An Overview of Technologies, Opportunities and Challenges". In: *IEEE Access* 9 (2021), pp. 26902–26925. DOI: 10.1109/ACCESS.2021.3053486.

[6] LASSE - UFPA. *Radio-Strike: A Reinforcement Learning Game for MIMO Beam Selection*. 2021. URL: https://aiforgood.itu.int/events/radio-strike-a-reinforcement-learning-game-for-mimo-beam-selection-in-unreal-engine-3-d-environments/.

[7] S. Wu, C. Wang, e M. Aggoune, M. M. Alwakeel, and X. You. "A General 3-D Non-Stationary 5G Wireless Channel Model". In: *IEEE Transactions on Communications* 66.7 (July 2018). Conference Name: IEEE Transactions on Communications, pp. 3065–3078. ISSN: 1558-0857. DOI: 10.1109/TCOMM.2017.2779128.

[8] J. Bian, C.-X. Wang, X. Gao, X. You, and M. Zhang. "A General 3D Non-Stationary Wireless Channel Model for 5G and Beyond". In: *IEEE Transactions on Wireless Communications* (2021). Conference Name: IEEE Transactions on Wireless Communications, pp. 1–1. ISSN: 1558-2248. DOI: 10.1109/TWC.2020.3047973.

[9] J. Shen et al. "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions". In: *https://arxiv.org/abs/1712.05884*. 2018.

[10] Theodore S Rappaport, Yunchou Xing, Ojas Kanhere, Shihao Ju, Arjuna Madanayake, Soumyajit Mandal, Ahmed Alkhateeb, and Georgios C Trichopoulos. "Wireless communications and applications above 100 GHz: Opportunities and challenges for 6G and beyond". In: *IEEE Access* 7 (2019), pp. 78729–78757.

[11] Cheng-Xiang Wang, Jie Huang, Haiming Wang, Xiqi Gao, Xiaohu You, and Yang Hao. "6G wireless channel measurements and models: Trends and challenges". In: *IEEE Vehicular Technology Magazine* 15.4 (2020), pp. 22–32.

[12] A. Klautau, P. Batista, N. Gonzalez-Prelcic, Y. Wang, and R. Heath. "5G MIMO Data for Machine Learning: Application to Beam-Selection using Deep Learning". In: *2018 Information Theory and Applications Workshop (ITA)*. Feb. 2018. URL: http://ita.ucsd.edu/workshop/18/files/paper/paper_3313.pdf.

[13] Muhammad Alrabeiah, Andrew Hredzak, Zhenhao Liu, and Ahmed Alkhateeb. "Viwi: A deep learning dataset framework for vision-aided wireless communications". In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE. 2020, pp. 1–5.

[14] A. Klautau, N. González-Prelcic, and R. W. Heath. "LIDAR Data for Deep Learning-Based mmWave Beam-Selection". In: *IEEE Wireless Communications Letters* 8.3 (June 2019), pp. 909–912. ISSN: 2162-2345. DOI: 10.1109/LWC.2019.2899571.

[15] Aldebaro Klautau and Nuria González Prelcic. "Realistic simulations in Raymobtime to design the physical layer of AI-based wireless systems". In: *ITU News MAGAZINE* 5 (2020), pp. 70–73.

[16] Aldebaro Klautau, Ailton de Oliveira, Isabela Pamplona Trindade, and Wesin Alves. "Generating MIMO Channels for 6G Virtual Worlds Using Raytracing Simulations (Submitted)". In: *2021 IEEE Statistical Signal Processing Workshop (SSP) (SSP 2021)*. Rio de Janeiro, Brazil, July 2021.

[17] Ailton Oliveira, Marcus Dias, Isabela Trindade, and Aldebaro Klautau. "Ray-Tracing 5G Channels from Scenarios with Mobility Control of Vehicles and Pedestrians". In: *XXXVII Simpósio Brasileiro de Telecomunicações e Processamento De Sinais - SBrT* (2019).

[18] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3&4 (Dec. 2012), pp. 128–138.

[19] Marcus Dias, Aldebaro Klautau, Nuria González-Prelcic, and Robert W Heath. "Position and LIDAR-aided mmWave beam selection using deep learning". In: *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)* (2019), pp. 1–5. DOI: 10.1109/SPAWC.2019.8815569.

[20] A. Ali, N. González-Prelcic, and R. Heath. "Millimeter Wave Beam-Selection Using Out-of-Band Spatial Information". In: *IEEE Transactions on Wireless Communications* 17.2 (2017), pp. 1038–1052. ISSN: 1536-1276. DOI: 10.1109/TWC.2017.2773532.

[21] V. Va, J. Choi, T. Shimizu, G. Bansal, and R. W. Heath. "Inverse Multipath Fingerprinting for Millimeter Wave V2I Beam Alignment". In: (2017). IEEE Early access. ISSN: 0018-9545. DOI: 10.1109/TVT.2017.2787627.

[22] N. González-Prelcic, A. Ali, V. Va, and R. W. Heath. "Millimeter-Wave Communication with Out-of-Band Information". In: 55.12 (Dec. 2017), pp. 140–146. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1700207.

[23] Cheng-Xiang Wang, Ji Bian, Jian Sun, Wensheng Zhang, and Minggao Zhang. "A survey of 5G channel measurements and models". In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3142–3168.

[24] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. "Airsim: High-fidelity visual and physical simulation for autonomous vehicles". In: *Field and service robotics*. Springer. 2018, pp. 621–635.

[25] A. Alkhateeb. "DeepMIMO: A Generic Deep Learning Dataset for Millimeter Wave and Massive MIMO Applications". In: *Proc. of Information Theory and Applications Workshop (ITA)*. San Diego, CA, Feb. 2019, pp. 1–8.

[26] REMCOM. *Wireless InSite*. 2019. URL: https://www.remcom.com/wireless-insite-em-propagation-software.

[27] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511807213.

[28] Isabela Trindade, Francisco Müller, and Aldebaro Klautau. "Accuracy Analysis of the Geometrical Approximation of MIMO Channels Using Ray-Tracing". In: *2020 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE. 2020, pp. 1–5.

[29] Felipe Bastos, Ailton Oliveira, João Borges, and Aldebaro Klautau. "Effects of Environment Model Complexity in Ray-Tracing simulation for UAV Channels". In: *X Conferência Nacional em Comunicações, Redes e Segurança da Informação* (2020).

[30] B. Li, Y. Shi, Z. Qi, and Z. Chen. "A Survey on Semantic Segmentation". In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2018, pp. 1233–1240. DOI: 10.1109/ICDMW.2018.00176.

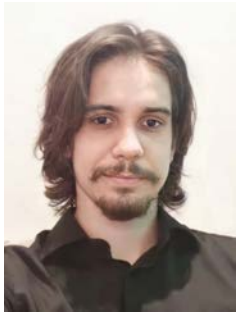
[31] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S. Lew. "A review of semantic segmentation using deep neural networks". In: *International Journal of Multimedia Information Retrieval* 7 (2018), pp. 87–93. ISSN: 1536-1276. DOI: https://doi.org/10.1007/s13735-017-0141-z.

[32] MIT Computer Vision team. *ADE20K dataset*. URL: http://groups.csail.mit.edu/vision/datasets/ADE20K/. (accessed: 04.07.2021).

[33] *Wireless InSite Reference Manual*. Version Version 3.3.0. Remcom Inc.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

AUTHORS



Ailton Oliveira is a B.Sc candidate in electrical engineering at Universidade Federal Pará, Brazil. He is currently a research student at the Telecommunications, Automation and Electronics R & D Center (LASSE/UFPA). His achievements were recognized with the outstanding undergraduate researcher award from LASSE/UFPA, and had an awarded article in 2020, by Brazilian Telecommunications Society (SBrT), with a research focused on machine learning applied to beam-selection. His current research interests include digital communications, 5G/B5G networks, MIMO systems, data science and machine learning.

Felipe Bastos received a technical degree in telecommunications from Instituto Federal do Pará (2015), Computer Engineer from Universidade Federal do Pará (2020) He also participated in an inter-university exchange at École Supérieure d'Informatique, Électronique, Automatique (ESIEA) through the BRAFITEC program, where he was an intern at the European Nuclear Research Center (CERN). Currently, he is with the Telecommunications, Automation and Electronics R & D Center (LASSE/UFPA) and pursues a Master of Science degree in electrical engineering at Universidade Federal do Pará (UFPA). He is interested in embedded systems, Internet of things, 5G networks, and telecommunication systems.

Isabela Trindade received a B.E. degree in electrical engineering from Universidade Federal do Pará, Brazil, in 2019. She is currently pursuing a M.Sc. degree in electrical engineering with the Telecommunications, Automation and Electronics R & D Center (LASSE/UFPA), under the supervision of Prof. Aldebaro Klautau. Her research interests include MIMO communications, channel modeling with ray tracing simulations and machine learning.

Walter Frazão is a B.Sc. candidate at Universidade Federal Pará, Brazil, Brazil. He has been a research student at the LASSE/UFPA since 2019 and a CNPq researcher at the Amazon Center for Excellence in Energy Efficiency (Ceamazon). He received an award in 2020 from the Brazilian Telecommunications Society (SBrT) for an article on machine learning applied to beam selection. His current research interests include 5G, MIMO communications and machine learning.

Arthur Nascimento started his degree in biomedical engineering in 2018 at the Universidade Federal do Pará. He is currently an undergraduate student researcher at LASSE/UFPA. He received an award for an article in telecommunications in 2020 and also worked on the regional organization of the 2020 ITU AI/ML in 5G Challenge. His current research interests include machine learning, artificial inteligence, computer vision and bioinformatics.

Diego de Azevedo Gomes received M.Sc. and Ph.D. degrees in electrical engineering (telecommunications) from Universidade Federal do Pará, Brazil, in 2012 and 2017, respectively. He is currently a professor at Institute of Geosciences and Engineering, Universidade Federal do Sul e Sudeste do Para (Unifesspa). He had two awarded articles in the year 2020, both regarding machine learning applied to beam selection. His current research interests include MIMO communications, digital signal processing, and machine learning.

Francisco Müller received his Bachelor's degree (2002), M.Sc. (2005) and Ph.D. (2010) in electrical engineering at Universidade Federal do Pará, Brazil. He has been an Associate Professor at Universidade Federal do Pará since 2011 and is associated with the 5G & IoT Research Group at LASSE/UFPA. He was a visiting scholar at Virginia Tech (2007). Current research interests include massive MIMO, 5G/6G channel modeling and estimation. He is a member of the IEEE Communications Society.

Aldebaro Klautau received the Bachelor's (UFPA, Brazil, 1990), M.Sc. (UFSC, Brazil, 1993) and Ph.D. degrees (University of California at San Diego, UCSD, 2003) in electrical engineering. Since 1996, he has been with UFPA and is now a full professor, the ITU-T Focal Point, and directs the 5G & IoT Research Group at LASSE/UFPA. He was a visiting scholar at Stockholm University, UCSD and The University of Texas at Austin. He is a senior member of the IEEE and a researcher of the Brazilian National Council of Scientific and Technological Development (CNPq).

INDEX OF AUTHORS

R

S

T

V

W

X

Y

Z

International
Telecommunication
Union

Telecommunication
Standardization Bureau (TSB)

Place des Nations
CH-1211 Geneva 20
Switzerland

ISSN: 2616-8375
Published in Switzerland
Geneva, October 2021