DATA-EFFICIENT GNN MODELS OF COMMUNICATION NETWORKS USING BETA-DISTRIBUTION-BASED SAMPLE RANKING

Max Helm[1], Benedikt Jaeger[1], Georg Carle[1]
[1]Department of Computer Engineering, Technical University of Munich, Germany

{helm, jaeger, carle}@net.in.tum.de

Abstract – *Machine learning models for tasks in communication networks often require large datasets to be trained. This training is cost intensive, and solutions to reduce these costs are required. It is not clear what the best approach to solve this problem is. Here we show an approach that is able to create a minimally-sized training dataset while maintaining high predictive power of the model. We apply our approach to a state-of-the-art graph neural network model for performance prediction in communication networks. Our approach is limited to a dataset of 100 samples with reduced sizes and achieves an MAPE of 9.79% on a test dataset containing significantly larger problem sizes, compared to a baseline approach which achieved an MAPE of 37.82%. We think this approach can be useful to create high-quality datasets of communication networks and decrease the time needed to train graph neural network models on performance prediction tasks.*

Keywords – Communication networks, data-centric AI, graph neural networks, latency, machine learning

1. INTRODUCTION

Machine learning, and specifically Graph Neural Networks (GNNs), play an important role in network performance modeling. One central indicator of network performance is the end-to-end latency of flows. Approaches such as RouteNet [1, 2, 3] can predict the mean end-to-end latency with small error margins. Creating a machine learning model requires a lot of computational resources, especially when we require it to scale to large problem sizes [4]. A large portion of this cost is located in the generation of the training dataset and training process itself. Methodologies that can reduce this cost are therefore of interest. The reduction of cost can be achieved, e.g., by a trade-off between training dataset size and model quality. Another approach is identifying and removing non-relevant or misleading samples, leading to a decrease in dataset size without negatively impacting model quality. We introduce such an approach consisting of two components and evaluate it on a network performance prediction task. We provide the following three contributions:

1. A methodology to generate a dataset containing edge cases and a methodology to reduce its size by ranking and removing samples;
2. application of this methodology to a state-of-the-art GNN model for network performance prediction (RouteNet [2]); and
3. comparison to a baseline approach.

We provide an overview of the background and related work in Section 2 and Section 3. The methodology is presented in Section 4 with an evaluation in Section 5 before concluding in Section 7. Digital artifacts are provided in Section 6.

2. BACKGROUND

This section provides an overview of the challenge, its requirements and restrictions, as well as basic information on GNNs and distribution types.

2.1 The challenge

This work was developed in the context of the Graph Neural Networking Challenge 2022 *Improving Network Digital Twins through Data-centric AI* [5, 6]. The goal of this challenge is to create a minimally-sized training dataset for a fixed GNN. The GNN is trained solely on this dataset and evaluated on an unknown, larger test dataset with significantly larger input problems. The quality of the generated dataset is determined by applying the trained GNN model on the test dataset and taking the Mean Absolute Percentage Error (MAPE). Furthermore, an evaluation dataset, following a similar distribution as the test dataset, was provided to locally test the solution.

2.2 Requirements and restrictions

The training dataset can contain at most 100 samples. The topology size is restricted to 10 nodes. A full list of requirements and restrictions can be found online[1]. The evaluation and test datasets contain topologies up to a size of 300 nodes. It was only possible to check the trained model against the test dataset for a total of 20 times. Therefore, we cannot rely on a brute-force approach, fuzzing approach, or tuning algorithms, e.g., grid search or Bayesian optimization [7]. Instead, we need to select training dataset samples carefully.

[1]`https://github.com/BNN-UPC/GNNetworkingChallenge/blob/2022_DataCentricAI/training_dataset_constraints.md`

2.3 Graph neural networks

Graph Neural Networks (GNNs) [8] are a machine learning approach working directly on graph-structured data. It takes advantage of spatial relations in data and the permutation invariance property of graphs, i.e., differently encoded isomorphous graphs lead to the same results. A graph is defined as a set G of vertices and edges as shown in Equation *(1)*.

$$G = (V, E) \tag{1}$$

Each vertex and edge can have an associated feature vector.
During the learning stage, a message passing and aggregation step is performed for n iterations. Each step consists of exchanging information between adjacent vertices, which is aggregated into a hidden state at each vertex by the aggregation function. The aggregation function is typically an invariant function.

2.4 Beta distribution

The beta distribution (β-distribution) is a family of distributions defined by two parameters. Depending on the parameter values, the distribution can take different shapes, e.g., approximations of a uniform, normal, exponential, gamma, or arcsine distribution. [9]
Due to this property, it is commonly used as a prior in Bayesian statistics. The multivariate generalization is the Dirichlet distribution. An extension of our approach could rely on this type of distribution to combine parameters.

3. RELATED WORK

This section provides an overview of related work in data-centric machine learning and applications of deep learning to performance modeling in communication networks.
Mirzasoleiman et al. [10] developed the method CRAIG to reduce the training dataset size while maintaining a very similar accuracy compared to the full training dataset. It works by selecting a subset of data that approximates the gradient of the full dataset as closely as possible. They report training speed-ups of up to $6\times$. We chose a different approach since we did not have access to a large, representative dataset.
Krishnateja et al. [11] provide a library that implements multiple core set selection methods, including CRAIG.
Hwang et al. [12] propose a solution for adding additional data to an existing dataset by sampling uniformly from a dimensionality-reduced distribution.
Coleman et al. [13] suggest a methodology to reduce dataset size based on a computationally efficient proxy model, decreasing the size by 50% without negative impact.
Xia et al. [14] propose an approach that should generalize to different complex, real-world datasets by utilizing a scoring based on the distance between a sample and its classes center.

Table 1 – Dataset sizes of different deep learning applications to performance estimation of communication networks

Author	Reference	Year	Training Dataset Size
Geyer and Bondorf	[18]	2019	100,000
Rusek et al.	[1]	2020	260,000
Geyer et al.	[19]	2021	54,000
Ferriol-Galmés et al.	[2]	2022	200,000
Afonso and Berton	[20]	2022	120,000
Our approach	—	2023	100

Cruz et al. [15] apply sample ranking to combat class imbalances. They differentiate between pointwise, pairwise, and listwise ranking.
Mazumder et al. [4] provide analyses for the importance of high-quality datasets, focusing on real-world impacts. They provide a tool to assess, among other things, the quality of datasets.
Eyuboglu et al. [16] propose a benchmark to evaluate data-centric machine learning approaches. They focus on three areas: cleaning training data on a budget, discover underperforming evaluation slices, and training data pruning (equivalent to core set selection).
Our approach differs from related work since we are able to synthetically create new data samples. Therefore, we present a combined approach consisting of both training data generation and core set selection.
For a more detailed survey on core set selection methods, the reader is referred to *Guo et al.* [17].
Performance modeling in communication networks with deep learning methods can be achieved using different approaches. There are approaches that combine network calculus with GNNs [18, 19], approaches that use GNNs to regress on mean delays [1, 2], and approaches that combine queuing theory and GNNs [20]. These approaches typically employ large training datasets to obtain good results. Table 1 shows a comparison of their respective dataset sizes compared to our approach.

4. METHODOLOGY

This section provides an overview of our approach, divided into three stages. Fig. 1 shows the three stages of dataset generation, sample ranking, and sample selection.

4.1 Initial dataset generation

The initial dataset generation consists of identifying which parameter value ranges can be derived from the evaluation dataset and which cannot be derived this way. An overview of the parameters and their chosen parameter ranges is shown in Table 2. Note that we decided on a fixed number of network nodes, graphs, and traffic matrices, i.e., we always have 100 different topologies of size 10 nodes with a unique traffic matrix each. This was done to ensure a maximum of training data as well as a maximal diversity of samples. Node and traffic parameter value ranges are taken directly from the values in the validation dataset, except for (a) one excluded link capacity due to challenge constraints, and (b) the number of on-off traffic

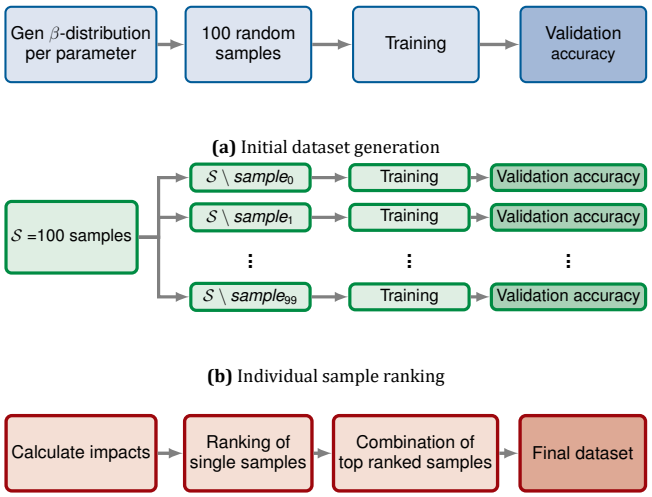


Fig. 1 – Three stages of the approach

instances which we increase from a fixed value of one to between zero and three.

The next step is to generate 100-sample datasets using these parameter value ranges. This is shown in Fig. 1 (a). The first step is to generate a β-distribution for each input parameter. This is done by picking the parameters of the β-distribution uniformly random as shown in Equation *(2)*.

$$\text{Beta}(\alpha \sim U(0.15, 5.0), \beta \sim U(0.15, 5.0)) \qquad (2)$$

Next, we create 100 random network configuration samples by sampling from the β-distributions of each parameter 100 times. An example of the node degree with different β-distribution parameters and resulting parameter value weights is shown in Table 3.

Next, we train the model on these 100 samples for a fixed budget of 20 epochs, leading to a validation accuracy for these 100 samples. We repeat this step n times to receive n datasets with associated validation accuracies. The β-distribution allows us to sample a diverse set of parameter values, because it can take on the shape of normally- or heavy-tailed distributions. Therefore, we not only try to replicate the validation dataset on a smaller scale but include edge cases of configurations. Such edge cases can be, for example, extremely high traffic load or unbalanced traffic types.

We use this step to generate n datasets of 100 samples each.

4.2 Individual sample ranking

The ranking of individual samples is repeated for each of the n datasets obtained in the previous step. We start with 100 samples of a single dataset. We generate 100 new datasets from this, with 99 samples each, by excluding one sample from each dataset, such that each sample is excluded exactly once. For example, we remove sample s_0 from the first dataset, sample s_1 from the second dataset, and sample s_{99} from the last dataset. We train the model with each of the newly-generated datasets, obtaining a validation accuracy for each of the datasets. Now, we can compute the impact of each sample on the original dataset. The impact I of one sample is defined as the relative error between the validation accuracy of the dataset where this sample was removed and the original 100 sample dataset as defined in Equation *(3)*, where $A(X)$ is the validation accuracy of training a model with dataset X. A larger impact means that this sample had a positive contribution to the overall model accuracy, whereas a negative impact means that the sample had a negative influence on the accuracy.

$$I(s_i) = (A(\mathcal{S} \backslash s_i) - A(\mathcal{S}))/A(\mathcal{S}) \qquad (3)$$

This results in an impact value that is normalized with respect to the different validation accuracies of each $\mathcal{S}_i$. We decided on this approach of comparing each sample to 99 samples at once because doing a full point-wise comparison is not feasible.

Table 2 – Parameters and their chosen value ranges

Parameter	Value
Network parameters	
Number of network nodes	10
Number of graphs	100
Number of traffic matrices	100
Node degrees	1,2,3,4,5,6,7
Node parameters	
Buffer size	val. data
Scheduler	val. data
Type of Service	val. data
Scheduling weights	val. data
Type of Service mapping	val. data
Link capacity	val. data (except 400k)
Traffic parameters	
Node parameters	val. data
Average bandwidth	val. data (783k values)
Packet sizes	val. data
Type of Service	0,1,2
Traffic	
Poisson	—
CBR	—
Num. On-Off instances	0,1,2,3
On period	2,3,4,5
Off period	2,3,4,5

Table 3 – Example node degree parameter value weights derived from β-distributions with different α and β parameter values

	Node degree	1	2	3	4
α	β	**Weights**			
0.1	0.2	0.125	0.997	0.906	0.081
1	1	0.124	0.55	0.541	0.941
1000	100	0.917	0.908	0.902	0.897

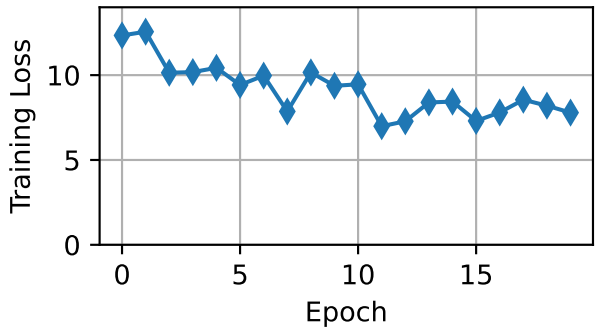


Fig. 2 – Training loss for Model 2 over 20 epochs

4.3 Sample selection

We collect the single samples and their impact scores from each of the n datasets, resulting in $100n$ datapoints. These samples are numerically ranked by their impact score. The best 100 samples, i.e., samples with the largest impact, are selected to form a new dataset $\mathcal{S}_{best}$ as shown in Equation *(4)*.

$$\mathcal{S}_{best} = \{s \in S | I(s_i) \geq I(s_p) \forall s_p \in S_p \wedge |S_p| = 100 \cdot (n-1)\} \quad (4)$$

5. EVALUATION

We compare the evaluation and test accuracy of our trained model to a provided baseline (BNN baseline) that was trained on 100 samples as well. We compare two of our models, Model 1 and Model 2. Model 1 is the best performing model generated using the β-distribution parameter value selection approach, without ranking indiviual samples. Model 2 is our final submission, which was generated by ranking and selecting single samples. Table 4 shows the results. We achieved a test MAPE score of 9.79%, decreasing the MAPE score almost four-fold from the baseline MAPE of 37.82%. A comparison between our two models shows an MAPE decrease from 10.20% without ranking to an MAPE of 9.79% with the sample ranking and selection strategy.

Fig. 2 shows the training loss of Model 2 over the full 20 epochs. We can observe that the best result is achieved after only 12 epochs. Training more epochs might reduce the loss further as indicated by the negative slope of a linear interpolation of the loss values. Fig. 3 shows the training loss distribution of all datasets before the sample ranking and selection process.

Taking a closer look at the topologies of the networks selected for the dataset used to train Model 2, we can observe a heterogeneous set of graphs. Three examples of topologically different graphs are shown in Fig. 4. Line-like topologies provide us with highly utilized bottleneck links as well as long flow paths. These are important since the model needs to generalize to significantly longer flow paths in the test dataset with networks of up to

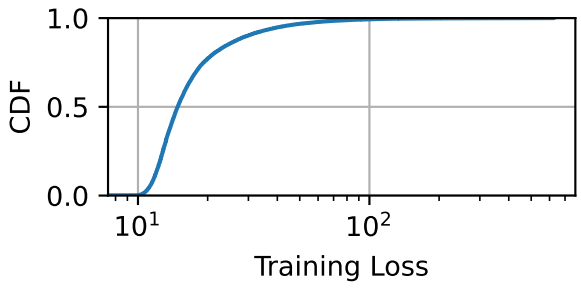
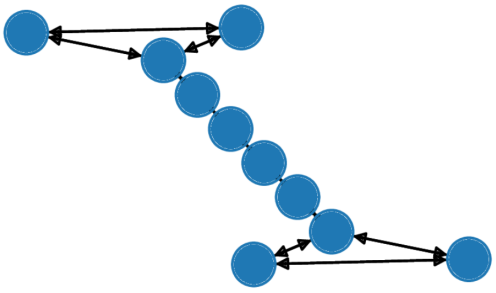
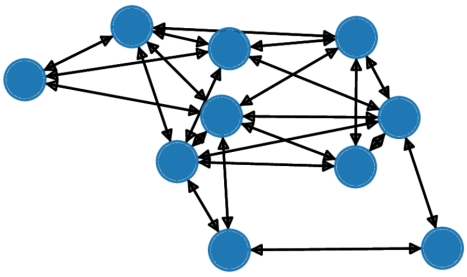


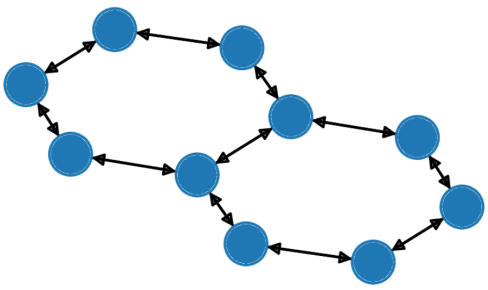
Fig. 3 – Training loss of all generated 100-sample datasets before sample ranking and selection



(a) Line-like network



(b) Densely connected network



(c) Double-ring network

Fig. 4 – Three topologically different networks from the dataset used to train Model 2

300 nodes. Densely connected networks provide us with many equally utilized links and a multitude of unique multiplexing points due to the high variance in node degrees. This is important to generalize well to larger networks since they have larger node degrees. Ring networks provide us with guaranteed circular dependencies between flows.

Table 4 – Comparison of the BNN baseline and our work on the validation and test dataset, using training dataset sizes of 100

Approach	Dataset Size	Val. MAPE	Test MAPE	Reference
BNN Baseline	100	26.52%	37.82%	[21]
Model 1 (after β-distribution sample generation)	100	7.37%	10.20%	—
Model 2 (after sample selection using ranking)	100	**6.99%**	**9.79%**	3^{rd} place at [5]

To further assess the influence of the sample ranking and selection, we take a look at graph theoretical properties of the generated and selected topologies in Table 5. We can see that the sample selection preferred graphs with more edges and a larger diameter. This makes sense since a larger diameter implies longer paths, which are important when scaling up to networks of 300 nodes. Following a similar logic, it also prefers networks with lower clustering coefficients. The same applies to the edge and vertex connectivity.

Table 5 – Comparison of graph metrics between the dataset used to train Model 2 and all datasets generated before ranking and selecting samples. Notation is as follows: Metric Model 2 (Metric before ranking).

Graph metric	Mean	Median
Number of edges	27.05 (18.83)	26.00 (20.00)
Diameter	4.52 (3.39)	4.00 (3.00)
Clustering coefficient	0.23 (0.39)	0.23 (0.41)
Edge connectivity	1.23 (1.77)	1.00 (2.00)
Vertex connectivity	1.20 (1.75)	1.00 (2.00)

For completeness, Table 6 lists all submissions with their respective approaches and test MAPE scores.

Table 6 – All submissions, their approaches, and their test MAPE scores

Submission	Approach	Model	Test MAPE
1	β-distribution	—	17.91%
2	β-distribution	—	12.58%
3	β-distribution	—	11.20%
4	β-distribution	—	11.07%
5	β-distribution	—	13.48%
6	β-distribution	—	10.95%
7	β-distribution	—	11.41%
8	β-distribution	—	11.45%
9	β-distribution	—	11.04%
10	β-distribution	Model 1	10.20%
11	β-distribution	—	10.45%
12	β-distribution	—	10.59%
13	β-distribution	—	10.59%
14	β-distribution	—	12.03%
15	β-distribution	—	10.72%
16	Sample ranking	—	10.04%
17	Sample ranking	Model 2	9.79%

6. REPRODUCIBILITY

We provide access to our trained Model 2, as well as to the dataset used to train it[2]. The dataset S_{best} consists of the 100 best-ranked samples.

7. CONCLUSION

We showed a method for generating minimally-sized datasets that can be used to train GNN models to a comparatively low error rate. Our approach consists of a combination of β-distribution-based parameter value selection and a leave-one-out sample ranking process. We showed that both parts of the approach have a measurable impact in terms of reduction of error rate. Overall, we were able to reduce the MAPE on a test dataset from a baseline of 38% to a MAPE of 9.79%. We think this approach can be useful to reduce the training time needed for GNN models of communication networks by removing unnecessary training samples from the process while still covering edge cases.

REFERENCES

[1] 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.

[2] Miquel Ferriol-Galmés, Krzysztof Rusek, José Suárez-Varela, Shihan Xiao, Xiang Shi, Xiangle Cheng, Bo Wu, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "Routenet-Erlang: A Graph Neural Network for Network Performance Evaluation". In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE. 2022, pp. 2018–2027.

[3] Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Krzysztof Rusek, Shihan Xiao, Xiang Shi, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "RouteNet-Fermi: Network Modeling with Graph Neural Networks". In: *arXiv preprint arXiv:2212.12070* (2022).

[4] Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Douwe Kiela, David Jurado, David Kanter, Rafael Mosquera, Juan Ciro, Lora

[2] `https://gitlab.lrz.de/gnnet-challenge-2022/dataset-100`

Aroyo, Bilge Acun, Sabri Eyuboglu, Amirata Ghorbani, Emmett Goodman, Tariq Kane, Christine R. Kirkpatrick, Tzu-Sheng Kuo, Jonas Mueller, Tristan Thrush, Joaquin Vanschoren, Margaret Warren, Adina Williams, Serena Yeung, Newsha Ardalani, Praveen Paritosh, Ce Zhang, James Zou, Carole-Jean Wu, Cody Coleman, Andrew Ng, Peter Mattson, and Vijay Janapa Reddi. *DataPerf: Benchmarks for Data-Centric AI Development*. 2022. DOI: 10.48550/ARXIV.2207.10062. URL:https://arxiv.org/abs/2207.10062.

[5] José Suárez-Varela. *Graph Neural Networking Challenge 2022 Improving Network Digital Twins through Data-centric AI*. Accessed: 2023-03-08. 2022. URL:https://bnn.upc.edu/challenge/gnnet2022/.

[6] José Suárez-Varela, Miquel Ferriol Galmés, Albert Lopez, Paul Almasan, Guillermo Bernárdez, David Pujol-Perich, Krzysztof Rusek, Loïck Bonniot, Christoph Neumann, François Schnitzler, François Taïani, Martin Happ, Christian Maier, Jia Lei Du, Matthias Herlich, Peter Dorfinger, Nick Vincent Hainke, Stefan Venz, Johannes Wegener, Henrike Wissing, Bo Wu, Shihan Xiao, Pere Barlet-Ros, and Albert Cabellos-Aparicio. "The Graph Neural Networking Challenge: A Worldwide Competition for Education in AI/ML for Networks". In: *CoRR* abs/2107.12433 (2021). arXiv: 2107.12433. URL: https://arxiv.org/abs/2107.12433.

[7] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization". In: *Advances in neural information processing systems* 24 (2011).

[8] 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.

[9] Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous Univariate Distributions, Volume 2*. Vol. 289. John wiley & sons, 1995.

[10] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. "Coresets for Data-efficient Training of Machine Learning Models". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6950–6960.

[11] Krishnateja Killamsetty, Dheeraj Bhat, Ganesh Ramakrishnan, and Rishabh Iyer. *CORDS: COResets and Data Subset selection for Efficient Learning*. Version v0.0.1. Mar. 2022. URL:https://github.com/decile-team/cords.

[12] Myunggwon Hwang, Yuna Jeong, and Wonkyung Sung. "Data Distribution Search to Select Core-set for Machine Learning". In: *The 9th International Conference on Smart Media and Applications*. 2020, pp. 172–176.

[13] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. "Selection via Proxy: Efficient Data Selection for Deep Learning". In: *arXiv preprint arXiv:1906.11829* (2019).

[14] Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, and Tongliang Liu. "Moderate Coreset: A Universal Method of Data Selection for Real-world Data-efficient Deep Learning". In: *The Eleventh International Conference on Learning Representations*. 2023.

[15] Ricardo Cruz, Kelwin Fernandes, Jaime S Cardoso, and Joaquim F Pinto Costa. "Tackling Class Imbalance with Ranking". In: *2016 International joint conference on neural networks (IJCNN)*. IEEE. 2016, pp. 2182–2187.

[16] Sabri Eyuboglu, Bojan Karlaš, Christopher Ré, Ce Zhang, and James Zou. "dcbench: A Benchmark for Data-Centric AI Systems". In: *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*. 2022, pp. 1–4.

[17] Chengcheng Guo, Bo Zhao, and Yanbing Bai. "Deepcore: A Comprehensive Library for Coreset Selection in Deep Learning". In: *Database and Expert Systems Applications: 33rd International Conference, DEXA 2022, Vienna, Austria, August 22–24, 2022, Proceedings, Part I*. Springer. 2022, pp. 181–195.

[18] Fabien Geyer and Steffen Bondorf. "DeepTMA: Predicting Effective Contention Models for Network Calculus using Graph Neural Networks". In: *Proceedings of the 38th IEEE International Conference on Computer Communications (INFOCOM 2019)*. INFOCOM 2019. Paris, France, Apr. 2019. DOI: 10.1109/INFOCOM.2019.8737496.

[19] Fabien Geyer, Alexander Scheffler, and Steffen Bondorf. "Tightening Network Calculus Delay Bounds by Predicting Flow Prolongations in the FIFO Analysis". In: *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. 2021, pp. 157–170.

[20] Bruno Klaus de Aquino Afonso and Lilian Berton. "QT-Routenet: Improved GNN Generalization to Larger 5G Networks by Fine-tuning Predictions from Queueing Theory". In: *arXiv preprint arXiv:2207.06336* (2022).

[21] Carlos Güemes. *Graph Neural Networking Challenge 2022 Quickstart Jupyter Notebook*. Accessed: 2023-03-09. 2022. URL:https://github.com/BNN-UPC/GNNetworkingChallenge/blob/2022_DataCentricAI/quickstart.ipynb.

AUTHORS

Max Helm obtained his M.Sc. degree from the Technical University of Munich in 2018, concentrating on communication networks. He is currently a Ph.D. student at the chair of Network Architectures and Services in the computer engineering department of the Technical University of Munich. His research areas include the application of graph neural networks, formal methods, and statistical methods to problems in the network performance modeling domain.

Benedikt Jaeger obtained his M.Sc. degree from the Technical University of Munich in 2018. He is currently a Ph.D. student at the chair of Network Architectures and Services in the computer engineering department of the Technical University of Munich. He is working on evaluating and modeling the performance of transport layer protocols such as TCP and QUIC.

Prof. Georg Carle holds the chair on Network Architectures and Services at Technical University of Munich (TUM). He conducts research on high-performance network technologies, network measurements, and network security. He studied electrical engineering at the University of Stuttgart. In 1996 he received his Ph.D. in computer science at University of Karlsruhe (now KIT). Subsequently he was postdoctoral scientist at Institut EURECOM, Sophia Antipolis, France, and at the Fraunhofer Institute for Open Communication Systems FOKUS, Berlin. In 2003, he joined University of Tübingen as a full professor, and in 2008 he accepted an offer from TUM.