ORACLE-BASED DATA GENERATION FOR HIGHLY EFFICIENT DIGITAL TWIN NETWORK TRAINING

Eliyahu Sason[1], Yackov Lubarsky[1], Alexei Gaissinski[1], Eli Kravchik[1], Pavel Kisilev[1]
[1]Toga Networks, Huawei Tel Aviv Research Center, 4 Haharash St., Hod Hasharon, Israel

NOTE: Main and corresponding authors: Eliyahu Sason, eli.sason@huawei.com and Yackov Lubarsky, yackov.lubarsky@huawei.com

Abstract – *Recent advances in Graph Neural Networks (GNNs) has opened new capabilities to analyze complex communication systems. However, little work has been done to study the effects of limited data samples on the performance of GNN-based systems. In this paper, we present a novel solution to the problem of finding an optimal training set for efficient training of a RouteNet-Fermi GNN model. The proposed solution ensures good model generalization to large previously unseen networks under strict limitations on the training data budget and training topology sizes. Specifically, we generate an initial data set by emulating the flow distribution of large networks while using small networks. We then deploy a new clustering method that efficiently samples the above generated data set by analyzing the data embeddings from different Oracle models. This procedure provides a very small but information-rich training set. The above data embedding method translates highly heterogeneous network samples into a common embedding space, wherein the samples can be easily related to each other. The proposed method outperforms state-of-the-art approaches, including the winning solutions of the 2022 Graph Neural Networking challenge.*

Keywords – Communication networks, deep learning, digital twin, graph neural networks, importance sampling

1. INTRODUCTION

In recent years, the networking community has shown a great interest in the development of Digital Twin (DT) technology for network performance modeling. The DT is an essential system for building computer network optimization tools [1, 2], such as optimizing network traffic (e.g. routing) given per-path performance (e.g. delay) to meet consumer SLA demands. The ultimate goal of the DT model is to deliver an exact digital replica of the real network, allowing it to mimic the underlying traffic behavior given real and what-if configurations. In this context, recent developments in Machine Learning (ML), and specifically in GNNs, offer a promising set of technologies for constructing an accurate DT with real-time performance. Typically, the quality of an ML-based model depends on two main factors: the model architecture and the training dataset. However, while the research community has shown significant progress in producing accurate and robust DT architectures [3, 4, 5], the data challenges have not yet been explored.

In particular, obtaining a good dataset from a real network is in practice a very challenging task due to technical constraints and legal regulations. Large-scale modern networks have hundreds or thousands of endpoints, claiming the demand of thousands of terabytes of data that need to be stored to generate sufficient datasets for model training. In addition, strict regulations on user privacy protection are being applied to network data, which prohibit or restrict network monitoring and the sharing of network activity records.

Finally, data collected from a real-world network will cover only a small portion of potentially important edge cases, such as network failures and highly congested network traffic scenarios. However, in order for the ML model to accurately model any network configuration and topology change, the dataset must contain a non-negligible amount of exemplars for such cases.

Another approach to obtain a highly representative dataset is to use one of the many available network simulation tools, such as OMNET++ [6] or ns-3 [7]. However, these tools have two significant drawbacks. First, data obtained through simulation may have distribution differences compared to real networks.
Second, high-quality simulation tools are often rather slow, especially when modeling large networks, and thus pose a significant resource limitation on the amount and size of data that can be modeled.

The Graph Neural Networking Challenge 2022 edition [8] addresses the fundamental data concern, how to generate a small and generalizable training dataset for the DT ML-based model. *The challenge objective is to produce a concise training dataset containing a limited amount of small simulated networks, such that the target GNN model generalizes and performs well on much larger networks*. Particularly, the training set should enable the GNN model to effectively scale to large network topologies, much larger than the small simulated samples seen during training, potentially having different traffic distribution. Participants are given a State-Of-The-Art (SOTA) GNN model, RouteNet-Fermi [4, 8], for network performance evaluation and a packet-level network simulator, OMNeT++ [6], to generate the training dataset.

In this paper, we present a clustering-based approach that aims to address this challenge. Our method is based on the concept of an *Oracle* model. In practice, this is a Routenet-Fermi model trained without the data budget limitation on a large number of simulated small networks.

The Oracle model serves two purposes: 1) by examining the Oracle performance on the validation set, we tune the data generation distribution to better match that of the hardest validation samples, and 2) it is used as a feature extractor of the generated samples. These features enable the comparison of the highly varying and complex samples in a common embedding space. In order to select the training dataset, we cluster all samples in the embedding space and select a small amount from each cluster. The presented approach achieves a state-of-the-art result, outperforming both our original method submitted to the GNN 2022 challenge [8] (group named 'Ghost Ducks') and the winning solution.

In summary, the main contributions of our work are as follows:

- We propose a clustering-based data sampling framework that allows a very efficient training of a Digital Twin (DT) network. This framework includes: (a) novel data generation approach that emulates the flow distribution of large networks using small networks, and (b) a clustering-based sampling method that finds the optimal training set. The small pool of highly important samples, obtained using our method, allows the DT model to effectively generalize to much larger networks while training on a handful of small-network examples, resulting in state-of-the-art performance.
- We propose a new data embedding method that maps the highly heterogeneous network samples into a common embedding space, wherein the samples can be related to each other in a meaningful way. While this embedding is used for clustering in our framework, it can also be utilized for other tasks.
- The newly proposed clustering method, uses multiple Oracle models to select the optimal centroid set for clustering the generated data samples. This allows us to efficiently limit the huge data pool of 270K simulated samples to a small pool of only 500 samples.

The remainder of this paper is organized as follows: First, we give an overview of the challenge problem statement and objectives in Section 2. We present related work in Section 3. We outline our pipeline using a clustering-based framework in Section 4. Finally, we present the results and summary in Section 5.

2. PROBLEM STATEMENT

2.1 Challenge description

The Graph Neural Networking Challenge 2022 [8] seeks to explore the impact of training data on GNN model performance, by placing limitations on the training dataset size and the properties of each sample. The challenge is to produce a training dataset, optimal for training a given GNN model with a triple constraint, the dataset size is limited to 100 samples, the topology of each sample must contain at most 10 nodes (the complete list of restrictions on the training set appears in Section 2.2) and finally, the training regime (e.g. number of epochs) for the given GNN architecture is fixed. The test data consists of samples with much larger topologies (50-300 nodes) to reference a realistic setup where the DT is trained in-house on small networks and then deployed on real topologies. The limitation on the topology size of the training samples presents a scalability and generalization challenge. The dataset size constraint, presents a generalization and convergence challenge while introducing a clear efficiency advantage of reducing training time and compute resources.

The organizers released an OMNeT++ [9] based simulator to generate the candidate training set samples. The model to be trained was the RouteNet-Fermi model [4], a GNN message-passing architecture. The inputs to the model consist of traffic matrix, topology, routing tables and a partial QoS specification (e.g. scheduling policy) of a given communication network. In the challenge, the model is tasked to predict the per-flow delay of the given network and the quality is measured using the Mean Absolute Percentage Error (MAPE) metric on the test set. According to the challenge rules, the training procedure is fixed to 20 epochs of 2000 steps, using the Adam optimizer [10] with a fixed learning rate of 0.1. A small validation set is provided, containing samples of large networks, similar to the test set distribution.

2.2 Data generation

2.2.1 Training data

Following the challenge description, the training dataset has to comply with different dataset constraints with regards to dataset size, graph topology, queues, and traffic flows.

The training dataset can contain a maximum of 100 unique samples, each composed of graph topology, routing table, and traffic matrix tuple.

The constraints on sample graph topology and nodes are as follows:

- Topology is limited to 10 nodes and must represent a connected graph with bidirectional links.

- The node queue scheduling policy can be one of four types: First In First Out (FIFO), Strict Priority (SP), Weighted Fair Queuing (WFQ), or Deficit Round Robin(DRR). All queue scheduling policies (except FIFO), use three queues with different priorities on nodes. Wherein the weights for WFQ and DRR define three different weight policies formed by three integers which sum up to 100.
- The node buffer size is constrained between 8000 and 64000 bits.
- Link bandwidth must be between 10000 and 400000 bps in multiples of 1000.

The limitations on generating traffic are as follows:

- Only one path is allowed between each source and destination pair.
- The average flow bandwidth must be between 10 and 10000 bps.
- The packet time distribution is constrained to one of three types: Poisson, CBR, ON_OFF.
- Packet size can take values between 256 and 2000 bits.
- Each flow Type of Service (ToS) priority value must be either 0, 1, or 2.

2.2.2 Validation and test data

The validation dataset, which consists of 130 samples with 50-300 nodes in the topology of each sample, was generated by the OMNeT++ [9] simulator. These topologies were artificially generated, based on the power-law out-degree algorithm [11]. The ranges of parameters α and β were taken from real-world data from the Internet Topology Zoo repository [12]. The scheduling policies were drawn randomly from the set of FIFO, SP, WFQ, DRR. Weights for each policy and the buffer sizes were randomly drawn as well.

A fluid model was chosen to calculate the values of link capacity, where there is a flow per each path transmitting the maximal defined throughput. In this model, the link capacity is defined as 4% of packet loss (average packet loss in the Internet is considered to be 2%-3% [13]). For each traffic matrix of each sample, the maximal average bandwidth value λ is sampled uniformly in the range of [400, 2000] bps. Then it is used to calculate the bandwidth for each source destination path, equal to $s * \lambda$, where $s \sim U[0.1, 1]$. The packet size distribution was randomly selected between five candidates, all having values of 500, 750, 1000, 1250, and 1500 bits but with different probability weights. The ToS for each flow was assigned randomly. The test dataset was generated in a similar manner as the validation dataset, only with a new set of topologies.

3. RELATED WORK

The creation of Digital Twin (DT) has recently attracted a lot of attention in the networking community. One of the most fundamental tasks for such a system is to estimate Key Performance Indicators (KPIs) given the network state, i.e. topology, device configuration and traffic matrix. Many types of network models aim to solve this challenge. These include analytical models based on queuing theory and Markov chains [14, 15]. Such systems can produce fast results but they are not accurate [16]. Another approach is to use a packet-level simulation tool, such as OMNET++ [9] or ns-3 [7]. These tools are accurate but come with a very high computational cost and slow running times.

3.1 Deep learning modeling

In recent years, there is a growing interest in Deep Learning (DL)-based network modeling [17], as these models are more accurate than analytical methods and are faster than the packet-level simulation tools. First works to leverage DL tools for KPI prediction were Deep-Q [18], utilizing the power of deep generative networks, and [19], using a fully-connected Neural Network (NN).

Current state-of-the-art modeling methods leverage the representation power and the generalization abilities of Graph Neural Networks (GNNs) [20]. In this paradigm, a network is naturally described as a graph $G = (V, E)$, where V are the nodes representing the network devices, and E are the edges, representing the network topology. An early work [21] used a GNN to predict the throughput of TCP flows.

The recently proposed RouteNet-Fermi model [4], is a SOTA GNN model for KPI prediction. In this model, the network is described by its topology, flow-level traffic, routing and interface-level configuration. The interface-level configuration allows us to model a queue scheduling policy for each device. The network model is represented by flows, queues and links that have circular dependencies between them. In order to solve this dependency the authors use a GNN-based message-passing model, where the dependencies are combined by several Recurrent Neural Networks (RNNs) [22].

XNEt [5] is a recent GNN-based networking model. In this work, the network is represented as a relation graph, and the cyclical dependency is solved with RNN, which is similar to RouteNet. This work allows us to obtain fine-grained temporal predictions by learning the state transition function between time steps.

From the provided overview, we can see that all current SOTA solutions that address the networking KPI prediction problem are deep learning-based models. In the proposed solution, we used RouteNet-Fermi [4] as the base model to present the effectiveness of our novel data generation and efficient training paradigm. This model was chosen as it is the current SOTA solution for the steady-state KPI prediction problem.

3.2 Data pruning

The OMNET++ [9] simulator can be used to create large amounts of training data. The data pruning line of research is trying to estimate which examples are useful for training and generalization of the model, and which are redundant or even hurt the trained model performance. This paradigm can be used to select the most beneficial samples out of a very large pool.
[23] introduced two methods to grade examples, and showed that taking only the hardest examples allows us training CIFAR-10 [24] on 50% of the original data without hurting performance. This article proposes two importance scores: 1) GraNd - the L_2 norm of the weight gradients on random initialized NN, and 2) EL2N - the L_2 norm of the classification error on a lightly pretrained NN. The data pruning objective can be reached with a clustering-based approach as shown in [25]. K-means clustering in the embedding space of a pretrained self-supervised model [26] is performed. The difficulty of each point is proportional to its distance to the nearest cluster centroid. One of their findings is that when the data is scarce, the easy samples should be retained instead of the hard ones. In addition, they provide a benchmarking study of ten other data pruning methods on the ImageNet dataset [27].
In summary, data pruning is an active research topic with applications developed mainly for the vision domain. In our work we follow the principles of [25] but adapt the data-pruning method to communication network modeling. First, we introduce a novel communication-network feature representation. Second, based on this representation, we develop a data selection method that filters a very large training data pool to extract samples most representative of the validation set, as described in Section 4.

4. OUR METHOD

In our approach, we use an *Oracle* model, a model trained without data budget limitation, in order to select the constrained training set. This model is utilized to guide the generation and selection process, as illustrated in Fig. 1.

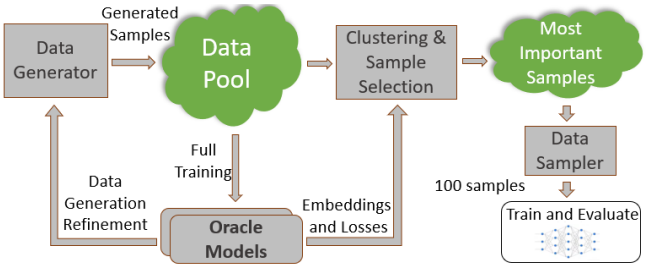

Fig. 1 – Our Solution pipeline

First, for the improvement of generated data distribution, by matching it to the validation distribution. Specifically, the Oracle highlights the hard validation samples, those with the highest loss. We enrich the data pool with data generated based on the statistics of these samples, as they may reveal under-represented scenarios.
This is an iterative process, achieved gradually through the following procedure:

- Step 1: Generate a dataset for training using the current data generation distribution and add to the data pool.
- Step 2: Train an Oracle model on the updated data pool.
- Step 3: Analyze the hard validation sample statistics.
- Step 4: Update the data generation distribution and repeat.

Second, we use the Oracle model to map the data samples to a common embedding space where each sample is represented by a fixed size vector, as illustrated in Fig. 2. While it is impossible to directly measure similarity between network samples due to their heterogeneous and highly complex data structure, we find that the distance measures within the embedding space given by the Oracle embeddings, hold information that is relevant in-context of the given task (i.e. KPI estimation).

In the final data selection step, we use these representations to cluster the generated samples against the validation samples according to their similarity. The clustered data pool allows us to obtain a very small dataset of highly important samples, by selecting a few samples from each group. The following sections describe our solution in more detail.

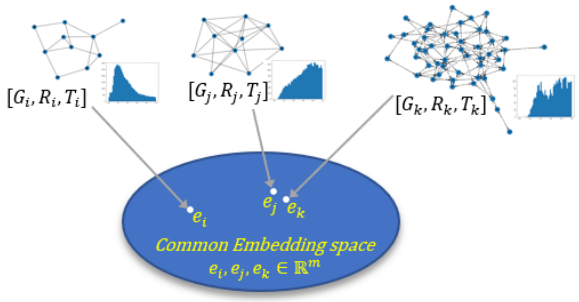

Fig. 2 – The problem of mapping highly complex and varying different samples composed from graph, routing and dynamic traffic denoted as G, R, T to common embedding space.

4.1 Distribution matching

The first goal of our data generation process was to produce a dataset that leads to a well-performing Oracle model, which performs similarly to the challenge organizers' claim of 5% MAPE on the validation dataset. Specifically, the challenge organizers trained the RouteNet-Fermi model with a large dataset, where each sample meets the challenge criterion, achieving an accurate per-path delay with 5% MAPE on the validation dataset.

Our data pool was created in several steps, that showed a steady improvement with respect to Oracle models' performance. We found that the trained models consistently perform badly on some validation samples. By analyzing the traffic KPIs of these "hard" validation samples, we gradually adjusted our data generation pipeline to generate samples with similar KPIs (e.g. heavy link loads), therefore improving the models' ability to address such cases.

Initially, we generated the data samples in a fully random fashion, uniformly sampling parameter values from the valid ranges (e.g. graph sizes, edge probability, queues policy, etc.). We fixed the routing to shortest path. We also focused on the larger graphs (sizes 6 and above) since those contain more entities (flows, queues, etc.) and more complex topologies.

Next, we limited our data generation to better match the statistics of the validation set. We found that many fields in the validation data have fixed or limited ranges of values, for that reason we updated our generated data distribution to sample exclusively from similar sets. This step improved our Oracle model compared to the fully random approach.

Finally, we generated datasets with hard samples. Namely, datasets where we tried to aggressively match the KPI of the generated samples – link loads, packet drop ratio and delay – to those in the hard validation samples. Ultimately, the hard samples typically represent networks with high link loads, high packet drops, and low delays.

In order to achieve this goal, we used several methods. To reduce delays we increased the link capacities and reduced the packet sizes. To increase the link loads we increased the overall traffic intensity in the generated networks. We also modified the routing algorithm, so that the routes it chooses are random paths between two nodes instead of the shortest path. This increases link utilization because each link is traversed by more traffic flows since each flow is potentially longer. To further increase the link utilization we generated samples where the link capacity is not random, but is derived from the total average bandwidth of the flows that pass through a link, such that link capacities tightly match the traffic demands. Specifically, in such data, the link capacity is set either to the nearest value or nearest value above the total bandwidth, from the given set of 10000, 25000, 40000, 100000, 250000, 400000, as illustrated in Fig. 4.

The KPI evolution of these three phases is illustrated in Fig. 3, showing a consistent statistical match convergence towards the hard validation samples. We denote by UNIFORM-C, UNIFORM-D and HARD the data generation distributions derived at each of these steps.

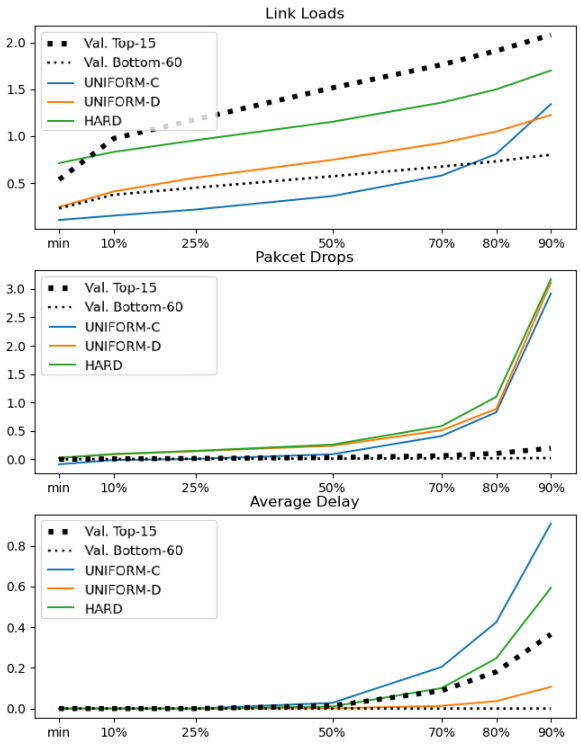

Fig. 3 – KPI comparison between the validation samples and the evolving generated datasets. The plots display KPI percentile values for each of the sets. Val.Top-15 denotes 15 highest loss validation samples. Val.Bottom-60 the 60 lowest loss 60 validation samples. UNIFORM-C, UNIFORM-D and HARD represent the three gradually improving data distributions described in Section 4.1.

4.2 Sample embeddings

The network data samples are highly complex heterogeneous entities, each consisting of different characteristics, e.g. a specific graph topology, queue policies, link bandwidths, routing and traffic patterns. In order to cluster the samples, we first need to find a suitable similarity metric by which to compare them. We define a shared space where samples can be compared one to another in a meaningful way, this process is illustrated in Fig. 5. This shared space is the space of fixed-size vector embeddings that we create for each of the samples, by running the sample data through the Oracle model and then extracting the resulting internal state tensors. Specifically, we extract the following internal tensors given by the RouteNet-Fermi architecture:

$PS \in \mathbb{R}^{F \times maxlen \times d}$: path state tensor
$LS \in \mathbb{R}^{L \times d}$: link state tensor
$QS \in \mathbb{R}^{Q \times d}$: queue state tensor
F: number of flows in a sample
L: number of links in a sample
Q: number of queues in a sample
$maxlen$: number of links in the longest flow path

These tensors are designed to represent the corresponding network entities. To obtain a fixed size representation for each of the tensors, we apply several pooling functions

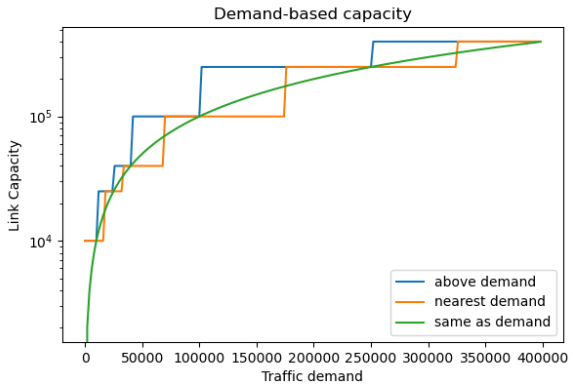

Fig. 4 – Demand based capacity. The link capacity is set according to the traffic demand along the link. We experimented with three flavors: "above demand" capacity is set to one of the fixed values nearest from above to the demand. "nearest demand" capacity is set to the nearest value from a fixed set. "same as demand" capacity is set exactly to the traffic demand value.

to each of the tensors along the entity dimension. We use min(), max() and mean() pooling. All the features are then concatenated to obtain a single fixed-size vector that contains rich data about the given sample. The full embedding generation process is described in the appendix, Section 6.1, Algorithm 1.

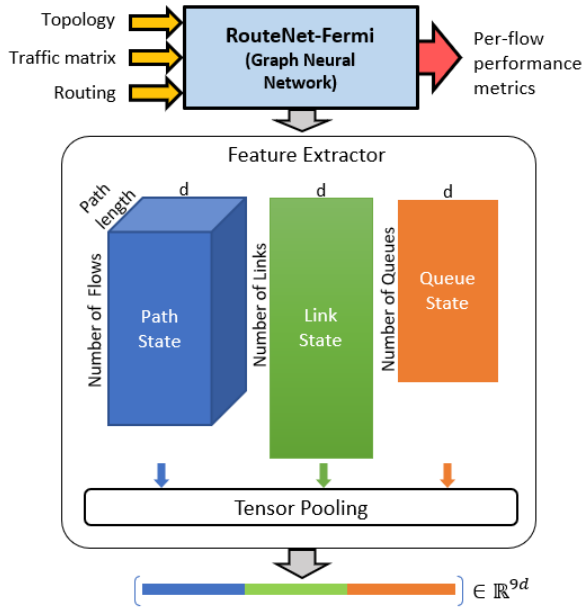

Fig. 5 – Illustration of sample embedding calculation

4.3 Clustering framework

Our clustering approach is a variant of centroid-based clustering models [28]. In order to relate the generated small-topology samples with those in the target distribution, the cluster center vectors are based on the validation samples. It is an open question as to how to define these. For example: get k centers by clustering validation samples themselves using k-means; split validation samples according to their loss, and then take more centers from difficult examples in order to give them an increased representation. Overall, in our experiments defining the centroids as the set of validation sample embeddings performed best compared to other methods; this is also the approach that is used in our final solutions.

While it is possible to use a single Oracle to define embeddings and cluster the generated data, we observe that Oracle models trained on different data distributions have slightly different strengths and weaknesses when evaluated on the validation data, as shown in Fig. 6. We take an inspiration from model averaging methods [29] and combine the embeddings from several Oracle models to obtain the best performing solution.

As discussed, initially the centroids are defined as the validation sample embeddings. Having more than one Oracle model means that we have several embedding spaces to be used. We cluster the data pool per each Oracle model. Every sample is assigned to its corresponding closest centroid, in terms of cosine distance in the embedding space. Given two Oracles, this step assigns all samples twice, namely to both sets of the centroids' embeddings. The determining final assignment in each cluster is according to the embedding space of the Oracle which achieved the lowest loss on the cluster validation sample.
This method may result in some validation clusters having no samples assigned to them. However, in our experiments, most validation samples have an assignment, which is a positive indication about the generated data quality. In addition, we find that using two Oracle models instead of one further helps to reduce the number of zero-size clusters.
After clustering the generated samples, we select a small amount of samples from each group. Namely, we take top-k samples from each cluster (k=3 or k=5), i.e. those samples that have the smallest distance to the cluster center, obtaining a very small pool of samples that relate to the validation data distribution in the embedding space. This allows us to efficiently limit the original pool of generated data to a much smaller dataset with highly important samples. During the competition, we generated a data pool of 270K samples and then reduced it to below 500 using this method. Finally, we randomly sample from this small pool clusters to get the final training set of 100 samples. The pseudo-code of our clustering and sampling logic is described in appendix, Section 6.1, algorithms 2 and 3.

5. EXPERIMENTAL RESULTS

5.1 Experimental setup

We verify our data-selection methods within the testing benchmark defined by the Graph Neural Networking Challenge 2022 (see Section 2 for details). The evaluation metric is MAPE, computed over per-flow delay predictions given by the trained RouteNet-Fermi model.

Fig. 6 – Loss comparison for the 130 validation-set samples between two Oracle models with validation MAPE of 6.15 (blue and 5.89 (orange). The x-axis indicates the sample indices, sorted by their loss in 5.89 Oracle.

First, we evaluate the Oracle model trained on selected distributions without a restriction on data budget. The model was trained for 2000000 iterations with a learning rate of 0.001 and a step scheduler with a factor of 0.75 every 150000 iterations.

In order to evaluate the final datasets selected by our method, we use the testing benchmark defined by the competition rules (see Section 2.1. The selected dataset of 100 samples of small networks is used to train the RouteNet-Fermi model under a fixed training regime defined by the competition organizers. The trained model is then evaluated on validation and test data consisting of large network samples.

Experiments are conducted on a Linux server with Intel(R Xeon(R Gold 6140 CPU @ 2.30GHz, 252GB RAM memory and one NVIDIA GeForce RTX 2080 Ti card used mainly for validation/test evaluation. To generate data, we use the Hydra configuration management library [30] to mix and match different configuration options at the command line.

5.2 Oracle model evaluation

In Table 1 we compare several Oracle models trained with the different data distributions specified in Section 4.1. UNIFORM-C denotes a model where the training data is sampled uniformly across the entire space of available configuration options. In UNIFORM-D, the training data is also sampled uniformly but in this case some of the parameters are restricted to a discrete set of values, taken from the validation set. Finally, the HARD data distribution represents "difficult" samples with higher link utilization and congestion scenarios. As the table shows, the Oracle result improves significantly when moving from UNIFORM-C to UNIFORM-D. This makes sense as the UNIFORM-D samples better match the validation and test set distributions. The best result of 8.16 MAPE is obtained when the samples are selected from the HARD dataset, representing more congestion-rich scenarios. This confirms our assumption that focusing the training on difficult heavy-traffic scenarios should get better result than mere random sampling, as those scenarios contribute significantly to the validation loss.

Table 1 – Oracle model performance

Data Distribution	MAPE (val.)	MAPE (test)
UNIFORM-C	7.3	10.10
UNIFORM-D	6.62	8.56
HARD	**5.89**	**8.16**

5.3 100-Sample model performance

We compare several data-selection methods to obtain the 100-sample set required by the challenge objective. Using each method we select 100 samples and then train the RouteNet-Fermi model according to the training regime given by the competition rules. The following methods are considered. RANDOM denotes the simplest baseline where the 100 samples used in training are chosen uniformly from the entire data pool of 270K samples. RANDOM-D denotes the configuration where samples are selected randomly from a smaller pool of 70K, generated by distributions that promote good accuracy in our Oracle models. In K-MEANS-EMBED, we first apply K-means clustering on the generated sample embeddings and then select equally from each of the clusters (nearest to the center) to reach a total of 100 samples. CHALLENGE-1st is the first place result of the competition (ours being the second). To the best of our understanding, they employ a rigorous analysis of the relationship between data parameters and the predicted KPIs and hand-design the samples for the training dataset. SINGLE-ORACLE-CLUST denotes our clustering solution, described in Section 4, using embeddings from a single Oracle model with 5.89 MAPE. TWO-ORACLE-CLUST denotes our competition solution. It uses combined embeddings from two Oracle models with MAPE scores of 6.15 and 5.89 as described in Section 4. We also add two variants where additional pooling functions are used in the embedding generation step, specifically percentile statistics. This should result in richer embeddings. The variants are denoted as SINGLE-ORACLE-CLUST-PCT and TWO-ORACLE-CLUST-PCT using one and two Oracles respectively, for clustering.

Table 2 summarizes the results. First we note that random methods perform far worse than those where samples rely on clustering. The RANDOM-D method outperforms RANDOM, indicating that data distributions that give rise to good Oracle models are also indispensable when trying to build a good small set for training. Second, combining two Oracle models improves the result compared to the single Oracle approach by 0.1 MAPE. The TWO-ORACLE-CLUST variant achieved the MAPE test score of 8.554 and a second place at the competition final ranking. Notably, the proposed TWO-ORACLE-CLUST-PCT achieves the best test score of 8.44, outperforming the competition top results. Evidently having richer features to construct the embeddings, contributed to better clustering of the samples. A visualization and more details about samples chosen for our final solution can be found in the appendix.

Table 2 – Model performance on selected 100 samples

Sample Selection Method	MAPE (val.)	MAPE (test)
RANDOM	8.56	12.34
RANDOM-D	7.08	9.62
K-MEANS-EMBED	6.65	8.68
CHALLENGE-1st	-	8.55
SINGLE-ORACLE-CLUST	6.53	8.70
SINGLE-ORACLE-CLUST-PCT	6.48	8.58
TWO-ORACLE-CLUST	6.31	8.55
TWO-ORACLE-CLUST-PCT	**6.30**	**8.44**

Table 3 – Data distributions composing the final solution set of 100 samples and the small post-clustering data pool of 398 samples from which the final solution was chosen. The numbers indicate the fraction of the set that was generated by the given data distribution.

Data Distribution	Fraction	
	100 samples	Post-Cluster pool
UNIFORM-C-SMALL	45%	38%
UNIFORM-C	10%	10%
UNIFORM-D	20%	23%
HARD	25%	29%

5.4 Qualitative analysis

We make a qualitative analysis of the final 100-sample set and the small post-clustering pool from which it was sampled. As most of the data is randomly generated, it is interesting to assess what kind of samples are considered by our selection process to be the most useful training data for the given task. To this end we break down these datasets by the data distribution used to generate each sample. The results are summarized in Table 3. To add more granularity to previously defined UNIFORM-C, UNIFORM-D and HARD distributions, we add UNIFORM-C-SMALL. UNIFORM-C-SMALL has similar uniformly distributed parameters as UNIFORM-C with the limitation that the topology is generated using a low edge probability of 0.1 which tends to generate tree-like topologies.

As the table shows, 25% of the final set and 29% of the pool are samples from HARD distribution. This confirms our suspicion that highly congested scenarios are a valuable resource for training. In addition, substantial 45% of the samples in the final set and 38% of the pool come from UNIFORM-C-SMALL distribution. Having few edges, the tree-like topologies have comparatively more scenarios with high link loads and congestion cases.

Fig. 7 compares the KPI measurements of the 100 samples in the final solution to that of the hardest and easiest samples from the validation set, grouped according to the sample-generation distributions. The packet drop statistics of the samples from UNIFORM, UNIFORM-C and HARD distributions match quite well those of the hard validation samples, indicating that perhaps this KPI more than others contributes to the success of optimizing the model. Link loads of samples selected from HARD and UNIFORM-D are at the higher range while those in UNIFORM-C and UNIFORM-C-SMALL are in the lower range, matching better to those of the low-loss validation samples. Delay values are higher than those in the validation set, possibly due to the link speed constraint given by the competition on the generated data, which is not present in the validation and test data.

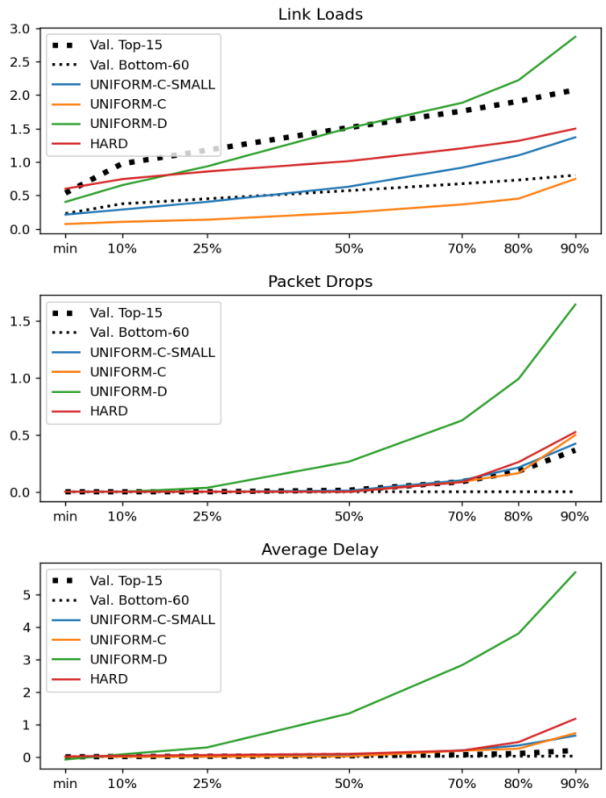

Fig. 7 – Percentile comparison of the samples in the best 100-sample solution dataset. The samples are grouped according to their generating distributions, UNIFORM-C, UNIFORM-C-SMALL, UNIFORM-D and HARD. Statistics for 15 highest-loss and 60 lowest loss validation samples are also shown for comparison.

6. CONCLUSION

This article introduced a novel approach for generating and sampling data for highly efficient training of network digital twin models, with a limited training budget and significant distribution differences between training and test data. It includes a novel data generation approach that emulates the flow distribution of large networks using small networks and a clustering-based sampling method that finds the optimal training set. The small pool of highly important samples, obtained using our method, allows the DT model to effectively generalize to much larger networks while training on a handful of small-network examples, resulting in state-of-the-art performance on the task posed at the 2022 Graph Neural Networking challenge.

The proposed method makes use of a new data embedding method that maps the highly heterogeneous network samples into a common embedding space, wherein the

samples can be related to each other in a meaningful way. While this embedding is used for clustering in our framework, it can also be utilized for other tasks.

In addition, we introduced a novel clustering method that uses multiple Oracle models to select the optimal centroid set for clustering the generated data samples. This allows us to efficiently limit the huge data pool of 270K simulated samples to a small pool of under 500 high-importance samples.

Overall, using the proposed method, we decomposed the task of generating a good dataset into a data generation phase and a data selection phase. In a realistic setup for constructing a network digital twin the development distribution (train) can be much different from the deployment distribution (test) due to network scale differences and other distinguishing factors. It is often not clear at the start which data distribution will be the most suitable for the training task. By using our method, data generation is a rather loose process whose main goal is to generate enough variability in a large data pool while the subsequent intelligent data-selection phase, ensures that the selected data is of a high quality necessary for the target task.

REFERENCES

[1] Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, et al. "Digital twin network: Opportunities and challenges". In: *arXiv preprint arXiv:2201.01144* (2022).

[2] Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, et al. "Network Digital Twin: Context, Enabling Technologies, and Opportunities". In: *IEEE Communications Magazine* 60.11 (2022), pp. 22–27.

[3] 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.

[4] 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: *IEEE/ACM Transactions on Networking* (2023).

[5] Mowei Wang, Linbo Hui, Yong Cui, Ru Liang, and Zhenhua Liu. "xnet: Improving expressiveness and granularity for network modeling with graph neural networks". In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE. 2022, pp. 2028–2037.

[6] Andras Varga. "OMNeT++". In: *Modeling and tools for network simulation* (2010), pp. 35–59.

[7] George F Riley and Thomas R Henderson. "The ns-3 network simulator". In: *Modeling and tools for network simulation* (2010), pp. 15–34.

[8] Barcelona Neural Networking Center at UPC. *Graph Neural Networking Challenge 2022 - Improving Network Digital Twins through Data-centric AI*. 2022. URL: https://bnn.upc.edu/challenge/gnnet2022/.

[9] András Varga. "Discrete event simulation system". In: *Proc. of the European Simulation Multiconference (ESM'2001)*. 2001, pp. 1–7.

[10] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[11] Christopher R Palmer and J Greg Steffan. "Generating network topologies that obey power laws". In: *Globecom'00-IEEE. Global Telecommunications Conference. Conference Record (Cat. No. 00CH37137)*. Vol. 1. IEEE. 2000, pp. 434–438.

[12] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. "The Internet Topology Zoo". In: *Selected Areas in Communications, IEEE Journal on* 29.9 (Oct. 2011), pp. 1765–1775. ISSN: 0733-8716. DOI: 10.1109/JSAC.2011.111002.

[13] S. Floyd and V. Paxson. "Difficulties in simulating the Internet". In: *IEEE/ACM Transactions on Networking* 9.4 (2001), pp. 392–403. DOI: 10.1109/90.944338.

[14] Giovanni Giambene. *Queuing theory and telecommunications*. Vol. 585. Springer, 2014.

[15] Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[16] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. "Experience-driven networking: A deep reinforcement learning based approach". In: *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE. 2018, pp. 1871–1879.

[17] Mowei Wang, Yong Cui, Xin Wang, Shihan Xiao, and Junchen Jiang. "Machine learning for networking: Workflow, advances and opportunities". In: *Ieee Network* 32.2 (2017), pp. 92–99.

[18] Shihan Xiao, Dongdong He, and Zhibo Gong. "Deep-q: Traffic-driven qos inference using deep generative network". In: *Proceedings of the 2018 Workshop on Network Meets AI & ML*. 2018, pp. 67–73.

[19] Albert Mestres, Eduard Alarcón, Yusheng Ji, and Albert Cabellos-Aparicio. "Understanding the modeling of computer network delays using neural networks". In: *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. 2018, pp. 46–52.

[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[21] Fabien Geyer. "Performance evaluation of network topologies using graph-based deep learning". In: *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. 2017, pp. 20–27.

[22] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. "Extensions of recurrent neural network language model". In: *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2011, pp. 5528–5531.

[23] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. "Deep learning on a data diet: Finding important examples early in training". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20596–20607.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". MA thesis. "University Of Toronto", ON, Canada, 2009.

[25] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S Morcos. "Beyond neural scaling laws: beating power law scaling via data pruning". In: *arXiv preprint arXiv:2206.14486* (2022).

[26] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised learning of visual features by contrasting cluster assignments". In: *Advances in neural information processing systems* 33 (2020), pp. 9912–9924.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[28] K Kameshwaran and K Malarvizhi. "Survey on clustering techniques in data mining". In: *International Journal of Computer Science and Information Technologies* 5.2 (2014), pp. 2272–2276.

[29] Gerda Claeskens, Nils Lid Hjort, et al. "Model selection and model averaging". In: *Cambridge Books* (2008).

[30] Omry Yadan. *Hydra - A framework for elegantly configuring complex applications*. Github. 2019. URL: `https://github.com/facebookresearch/hydra`

AUTHORS

Eliyahu Sason received his B.Sc. and M.Sc. degrees in electrical and electronics engineering from the Technion, Israel Institute of Technology, in 2013 and 2018 respectively. He currently serves as an AI researcher at Huawei Tel Aviv Research Center, Cognitive Computing Lab. Previously he worked at IBM Watson Research Labs. His research interests include deep neural networks and their application in computer networking and computer vision domains. Recently, he focuses on network digital-twin modeling.

Yackov Lubarsky received his B.Sc. and M.Sc. degrees in computer science from Tel Aviv University (TAU). Currently, he is an AI researcher at Huawei Tel Aviv Research Center, Cognitive Computing Lab. His research interests include deep neural network compression methods, time series analysis, graph neural networks, natural language processing and deep learning applications in communication networks.

Alexei Gaissinski received his B.Sc. and M.Sc. degrees in electrical and electronics Engineering from Tel Aviv University (TAU), in 2012 and 2016 respectively. He is currently an AI researcher at Huawei Tel Aviv Research Center, Cognitive Computing Lab. His research interests are focused on compression of neural networks and graph neural networks applied in the networking domain.

Eli Kravchik received his B.Sc. in biomedical engineering cum laude, from the Technion, Israel Institute of Technology in 2010, as well as an M.Sc. in electrical engineering in 2015, and a B.A in mathematics in 2015. His fields of interest are NN quantization, GNN compression,

reinforcement learning, active learning, and distributed training of NNs. Currently he is an AI researcher at Huawei Tel Aviv Research Center, Cognitive Computing Lab.

Pavel Kisilev is Cognitive Computing Lab Director at Huawei Tel Aviv RC. He received a PhD in computer science from Technion, IIT and has held various senior and lead research positions at Hewlett Packard and IBM Watson Research Labs. He has 80+ patents and 75+ published papers and book chapters.

APPENDIX

6.1 Algorithms

Algorithm 1 Create sample embedding

$PS \in \mathbb{R}^{F \times maxlen \times d}$: path state tensor
$LS \in \mathbb{R}^{L \times d}$: link state tensor
$QS \in \mathbb{R}^{Q \times d}$: queue state tensor
$n \in \mathbb{R}^{F}$: vector containing number of steps in every flow path
F: number of flows in a sample
L: number of links in a sample
Q: number of queues in a sample

/* aggregate flow features along the flow path dimension */
Function Calculate_embeddings(PS, LS, QS)
$\forall f : 1..F, p_f \leftarrow \frac{1}{n_f}\sum_{j=1}^{n_f} PS_{f,j}$
$PS \leftarrow stack(p_f, dim = 0)$ /* this creates tensor $PS \in \mathbb{R}^{F \times d}$ */
$PS_{emb} \leftarrow embed\_min\_max\_mean(PS)$
$LS_{emb} \leftarrow embed\_min\_max\_mean(LS)$
$QS_{emb} \leftarrow embed\_min\_max\_mean(QS)$
$e \leftarrow concat([PS_{emb}, LS_{emb}, QS_{emb}])$ /* resulting vector $e \in \mathbb{R}^{9d}$ */
return e

/* helper function */
Function embed_min_max_mean(X)
return $concat([min(X, dim = 0)], max(X, dim = 0), mean(X = 0))$

Algorithm 2 Assign clusters

n_{ora}: number of Oracle models
k: number of samples to keep in each cluster
V: number of validation samples
G: number of generated samples
d: size of the embedding vectors
$GenEmb^{(1)}, .., GenEmb^{(n_{ora})} \in \mathbb{R}^{G \times d}$: embeddings of generated samples from different Oracles
$ValEmb^{(1)}, .., ValEmb^{(n_{ora})} \in \mathbb{R}^{V \times d}$: embeddings of the validation samples from different Oracles
$ValLoss^{(1)}, .., ValLoss^{(n_{ora})} \in \mathbb{R}^{V}$: validation sample losses from different Oracles

$D \leftarrow [0]_{G \times V}$
for $v \leftarrow 1$ to V **do**
  /* Select the Oracle that performs best */
  /* on the given validation sample */
  $ora \leftarrow \underset{i \in 1..n_{ora}}{argmin}(ValLoss^{(i)}[v])$
  /* calcualte embeddings similarity */
  for $g \leftarrow 1$ to G **do**
    $D_{g,v} \leftarrow CosineDist(GenEmb^{(ora)}[g], ValEmb^{(ora)}[v])$
  end for
end for

for $v \leftarrow 1$ to V **do**
  $CLS_v \leftarrow \{g \in 1..G : \text{argmin}_i(D[g, i] = v)\}$
end for

/* Compute Top-k samples in each cluster */
/* nearest to the cluster validation sample */
for $v \leftarrow 1$ to V **do**
  $Sorted_v \leftarrow$ sort $g \in CLS_v$ in increasing order of $D_{g,v}$
  $Top_v \leftarrow Sorted_v[1..V]$
end for

return $\{Top_v$ for v in $1..V\}$

Algorithm 3 Sample from clusters

V: number of validation samples
$\{Top_v : v \in 1..V\}$: Top-k samples in each of the v clusters
b: size of the output dataset

$S \leftarrow \emptyset$
while $|S| < b$ **do**
  $remainder \leftarrow b - |S|$
  $u \leftarrow random_sample(Top_v\, S, 1) for v \in 1..V$
  if $|u| < remainder$ **then**
    $u \leftarrow random_sample(u, remainder)$
  end if
  $S \leftarrow S \cup u$
end while

return S

6.2 Final solution samples

Fig. 8 shows the topologies selected by our algorithm for the final solution. All graphs are generated with the Erdos-Renyi method, using 8-10 nodes and edge probabilities between 0.1 and 0.4. As mentioned in Section 5, two data distributions account for a large part of the final solution: UNIFORM-C-SMALL and HARD.

In UNIFORM-C-SMALL the edge probability for graph generation is 0.1 and the rest of the parameter values are sampled uniformly from the valid ranges of each parameter.

In the HARD data distributions the following are used to set the parameters:

- Link capacities are set either according to the traffic demand from the set of 10000, 25000, 40000, 100000, 250000, 400000 bps.
- The scheduler is selected randomly with equal probability from all the available choices - FIFO, SP, WFQ and DRR.
- Buffer sizes are sampled using a weighted choice out of the list [8000, 16000, 32000, 64000] bits with the respective weights 0.33, 0.24, 0.23, 0.2.
- WFQ and DRR weights are set randomly to one of the following ratios: [33.3, 33.3, 33.4], [50.0, 40.0, 10.0], [60.0, 30.0, 10.0], [70.0, 25.0, 5.0], [90.0, 5.0, 5.0].
- Average bandwidth for each flow is sampled uniformly in the range of [2000, 6000] bps.
- Time of service is set to 1, 2 or 3 with respective weighs 0.1, 0.3 and 0.6.
- Packet sizes are sampled using a weighted choice out of [500, 750, 1000, 1250, 1500] bits with respective weights of [0.53, 0.16, 0.07, 0.1, 0.14].
- The packet time distribution is uniformly chosen from the three available types. For on/off distribution the on and off periods are both set to 5 seconds (similar to the validation dataset).

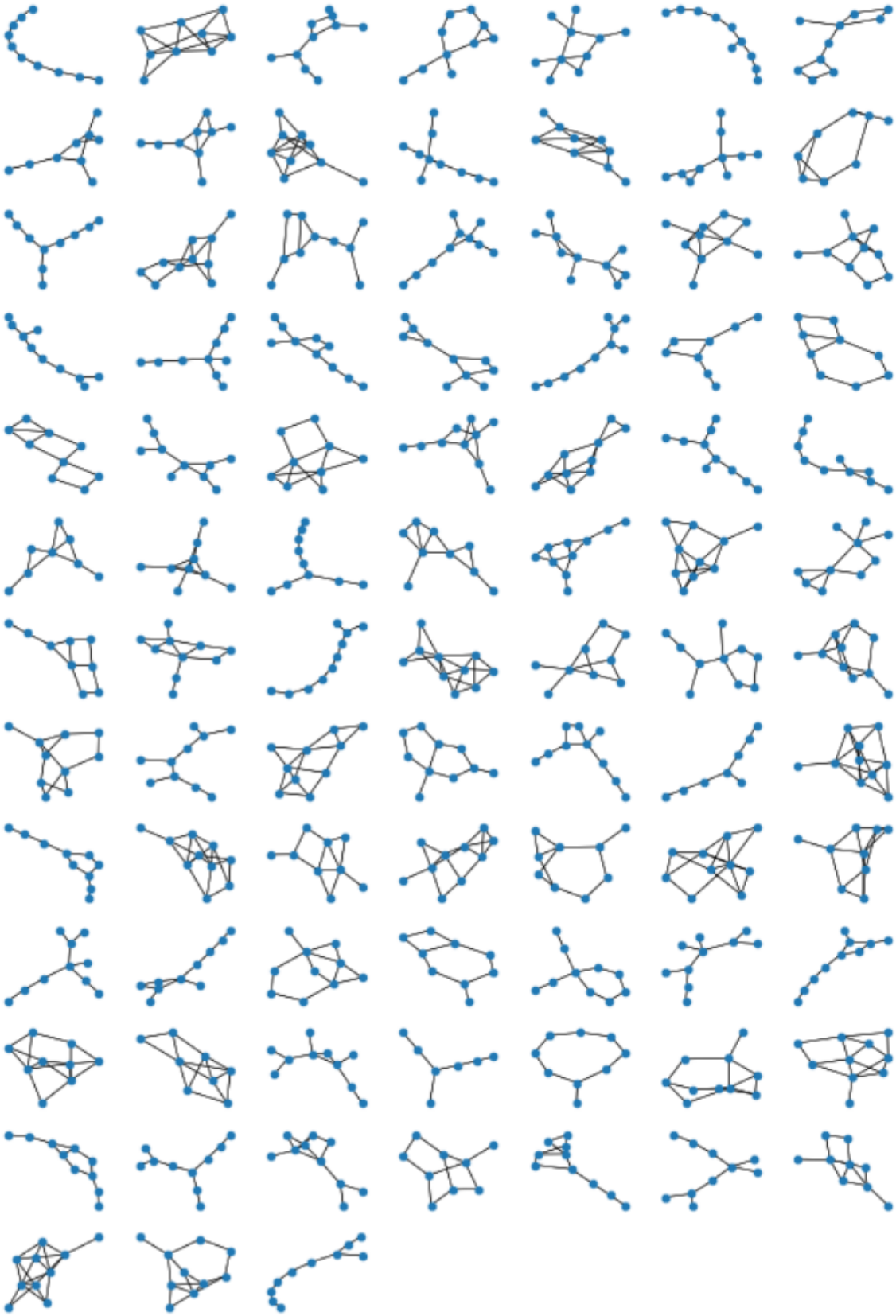

Fig. 8 – Topologies of the samples in the final solution. Out of the 100 samples we only show the 87 unique non-isomorphic topologies.