Graph Neural Networking Challenge 2022

Improving Network Digital Twins through Data-centric AI

Team: GhostDucks

Eli Sason, Eli Kravchik, Alexei Gaissinski, Yackov Lubarsky

toganetworks a HUAWEI company

Barcelona Neural Networking Center

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

AI for Good Machine Learning in 5G Challenge

Applying machine learning in communication networks

International Telecommunication Union (ITU)

ITU AI/ML in 5G Challenge

Motivation: Digital Twin

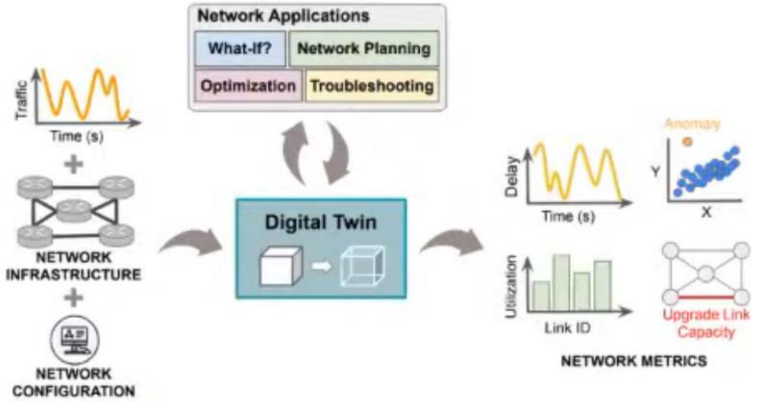
- A Network Digital Twin is a virtual replica of a physical network

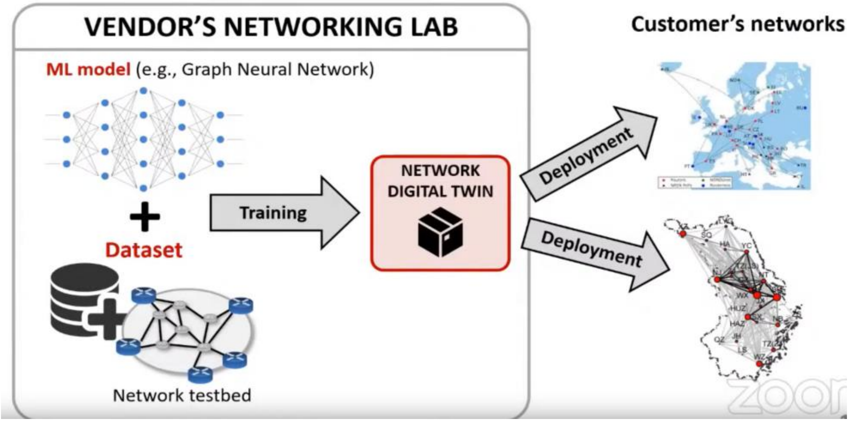
- In enables to reproduce the network behavior under certain what-if scenarios
  - What happens if change in configuration?
  - What happens if there is a random failure?
  - What is the best upgrade given limited budget?
  - Can the network support new SLAs?

- ML-based Digital Twin

  A good training dataset is crucial for achieving good performance
  - Training with data from real network is not feasible
    - Requires to cover edge cases
    - Recording traffic on per-flow basis is costly and may slow down equipment
    - Privacy concerns may prohibit access to historical records
  - Using simulated data is not feasible when network size is very large, because available simulation tools are very slow
  - Need for ML models that generalize to other networks, not seen during training
    Especially, to networks that are much larger than the ones in the train set

How To Produce a Good Dataset?





ML based Digital Twin

Motivation: Digital Twin

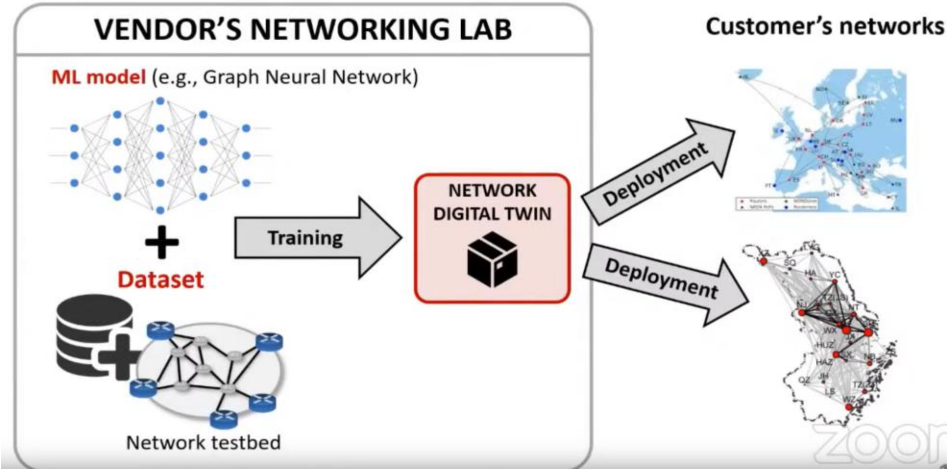
How To Produce a good Dataset?
There is no research on how to produce good datasets for ML models applied to networking!

A good dataset requires…

- Domain expert knowledge → understanding are relevant features to the ML model
- Good coverage of possible cases (e.g. congestion levels)
- Edge cases (e.g. different types of failures)
- Avoid unambiguous labels (e.g. noise)
- Compact → to save training times and costs

Potential benefits:

- **Large performance gains** (better coverage of important training samples)
- **Cost savings (**less training samples needed)



ML based Digital Twin

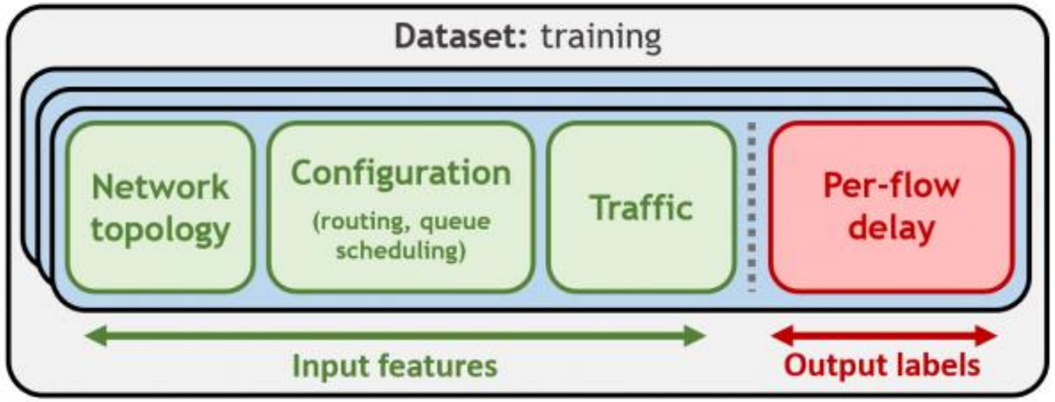
Graph Neural Networking Challenge 2022

Overview

- In recent years, the networking community has produced robust Graph Neural Networks (GNN) that can accurately mimic complex network environments. Modern GNN architectures enable building lightweight and accurate Network Digital Twins that can operate in real time. However, the quality of ML-based models depends on two main components: the model architecture, and the training dataset. In this context, very little research has been done on the impact of training data on the performance of network models.
- This edition of the Graph Neural Networking challenge focuses on a fundamental problem of current ML-based solutions applied to networking: how to generate a good dataset. We invert the format of traditional ML competitions, which follow a model-centric approach. Instead, we propose to explore a data-centric approach for building accurate Network Digital Twins.

Problem Statement

- Participants will be given a state-of-the-art GNN model for network performance evaluation (RouteNet-Fermi), and a packet-level network simulator to generate datasets. They will be tasked with producing a training dataset that results in better performance for the target GNN model.
- The training dataset should help the GNN model scale effectively to samples of larger networks than those seen during training.



Challenge Constraints

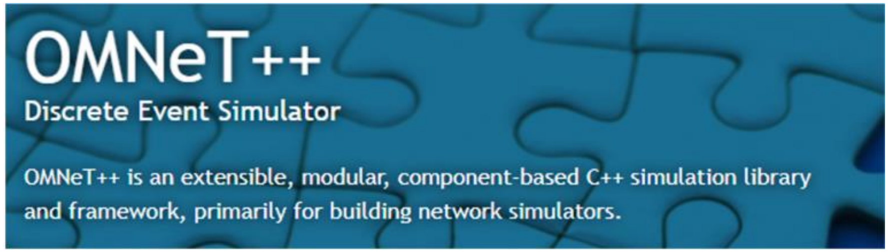
Simulation Parameters

1. Topology
- Network Graph (nodes, edges)
- Link Capacity
- Per Node:
  - Scheduling Policy: FIFO, SP, WFQ, DRR
  - Buffer size
  - WFQ/DRR weights

2. Routing Table

3. Traffic Matrix
- Avg. bandwidth
- TOS – type of service (QOS)
- Packet time distribution
- Packet size distribution

Packet-level network simulator based on OMNeT++



Computationally very expensive to simulate real networks at scale.
The cost is proportional to the number of packet events in the network.

Simulation Output
- Link Loads
- Flow Delays
- Packet Drops
- etc.

"To make the problem tractable, we have scaled it down. Participants will be able to quickly generate their training datasets on commodity hardware. We put the constraint that **training datasets must have a maximum of 100 samples, and these samples must be from small networks of up to 10 nodes.**"

Small networks
- Up to 10 nodes
- Max 90 flows

Tight Budget
- Up to 100 samples

Train set
(to find)

Train →

RouteNet-Fermi
(Graph Neural Network)

Fixed training regime

Test on →

Large Networks
- Up to 300 nodes
- Max 89700 flows

Val and Test sets
(given)

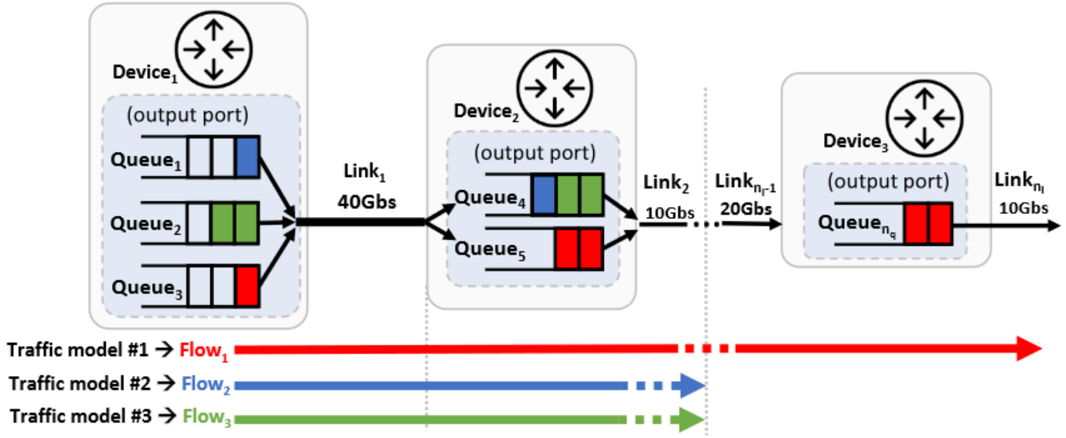
RouteNet

- A Message Passing GNN
- Input: Network configuration, Traffic demands
- Output: per-flow KPI
- Generates internal states for each flow, queue, link
    - State of *flows* is affected by the state of queues and links along the flow
    - State of *queues* depends on flows passing through them
    - State of *links* depends on the state of queues that inject traffic into the link and their scheduling policy

$$h_{f_i} = G_f(h_{q^{i,1}}, h_{l^{i,1}}, \ldots, h_{q^{i,M}}, h_{l^{i,M}})$$

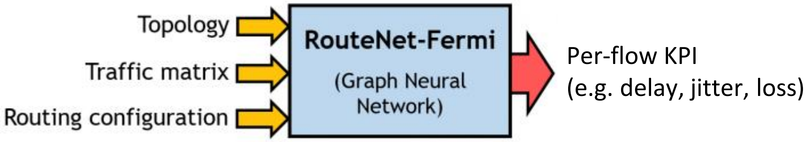
$$h_{q_j} = G_q(h_{f_1}, \ldots, h_{f_I}), \quad f_i \in Q_f(q_j)$$

$$h_{l_k} = G_l(h_{q_1}, \ldots, h_{q_J}), \quad q_j \in L_q(l_j)$$



- Scales to larger graphs, via normalizing by link capacities

$$x_{l_{load}} = \frac{1}{x_{l_c}} \sum_{f \in L_f(l_j)} \lambda_f \qquad \hat{d}_q = \frac{R_{f_d}(h^T_{f,l})}{x_{l_c}} \qquad \hat{d}_t = \frac{x_{f_{ps}}}{x_{l_c}} \qquad \hat{d}_{link} = \hat{d}_q + \hat{d}_t \qquad \hat{y}_{f_d} = \sum_{link \in f} \hat{d}_{link}$$



Algorithm 1 Internal architecture of RouteNet-F.

Input: $\mathcal{F}$, $\mathcal{Q}$, $\mathcal{L}$, x_f, x_q, x_l
Output: $\hat{y}_{f_d}$

```
for each f ∈ F do h_f^0 ← HS_f(x_f)
for each q ∈ Q do h_q^0 ← HS_q(x_q)
for each l ∈ L do h_l^0 ← HS_l(x_l)

for t = 0 to T-1 do                                   ▷ Message Passing Phase
    for each f ∈ F do                                 ▷ Message Passing on Flows
        Θ([·,·]) ← FRNN(h_f^t, [·,·])                 ▷ FRNN Initialization
        for each (q,l) ∈ f do
            h_{f,l}^t ← Θ([h_q^t, h_l^t])             ▷ Flow: Aggr. and Update
            m̃_{f,q}^{t+1} ← h_{f,l}^t                 ▷ Flow: Message Generation
        h_f^{t+1} ← h_{f,l}^t
    for each q ∈ Q do                                 ▷ Message Passing on Queues
        M_q^{t+1} ← Σ_{f∈Q_f(q)} m̃_{f,q}^{t+1}        ▷ Queue: Aggregation
        h_q^{t+1} ← U_q(h_q^t, M_q^{t+1})             ▷ Queue: Update
        m̃_q^{t+1} ← h_q^{t+1}                         ▷ Queue: Message Generation
    for each l ∈ L do                                 ▷ Message Passing on Links
        Ψ(·) ← LRNN(h_l^t, ·)                         ▷ LRNN Initialization
        for each q ∈ L_q(l) do
            h_l^t ← Ψ(m̃_q^{t+1})                      ▷ Link: Aggr. and Update
        h_l^{t+1} ← h_l^t
for each f ∈ F do                                     ▷ Flow: Readout
    ŷ_{f_d} = 0                                       ▷ Initializing the flow delay
    for each (q,l) ∈ f do
        d̂_q = R_{f_d}(h_{f,l}^T)/x_{l_c}              ▷ Queueing delay
        d̂_t = x_{f_ps}/x_{l_c}                        ▷ Transmission delay
        d̂_link = d̂_q + d̂_t
        ŷ_{f_d} = ŷ_{f_d} + d̂_link                    ▷ Sum of link delays along the flow
```

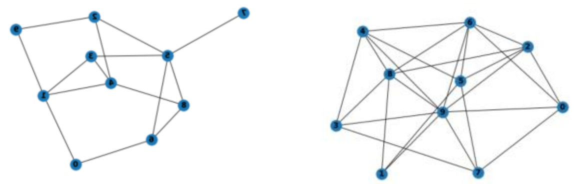
K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, A. Cabellos-Aparicio, "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN," In Proceedings of the ACM Symposium on SDN Research (SOSR), pp. 140-151, 2019. [ACM SOSR 2019] [arXiv]

Data Constraints

<u>Topology</u>		
Graph Size	5-10 nodes	Larger more complex, but want generalization
Edge probability	p=0.1..0.5	Larger -> more paths, longer paths
Link Bandwidth	10K-400K	Larger -> smaller delays, less congestion
Scheduling Policy	FIFO/SP/WFQ/DRR	
Buffer Size	8K-64K	Larger -> longer delays, sometimes less drops
<u>Routing</u>		
	Shortest Path	
	Random Path	Longer paths, more delay and link loads
	Shortest by link bw	
<u>Traffic</u>		
Average bandwidth	10-10000	Main traffic load control
TOS		
Packet Time Distr.		
Packet Size Distr.		Smaller -> reduce delay

Generated Samples - 10 nodes

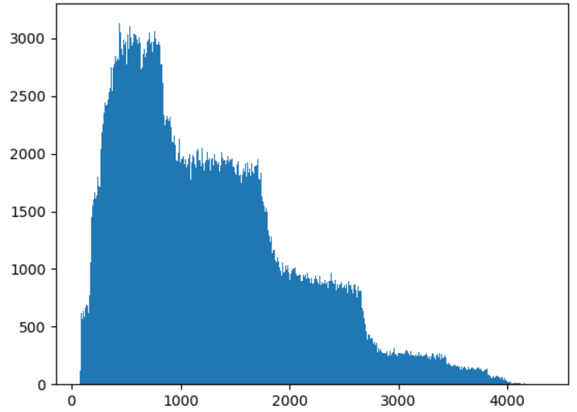
P=0.3 P=0.5



Val. Sample - 50 nodes

```
Route Lengths
length: 2       count: 103776   ratio: 0.022
length: 3       count: 437098   ratio: 0.091
length: 4       count: 1315954  ratio: 0.275
length: 5       count: 1794118  ratio: 0.375
length: 6       count: 879294   ratio: 0.184
length: 7       count: 207656   ratio: 0.043
length: 8       count: 35016    ratio: 0.007
length: 9       count: 4624     ratio: 0.001
length: 10      count: 456      ratio: 0.000
length: 11      count: 48       ratio: 0.000
Symmetric routes count: 3313726
Asymmetric routes count: 1464314
Packets Time Distribution
Poisson         0.333
CBR             0.333
ON-OFF          0.334
ON-OFF Times Distribution
5.0, 5.0        1.0
ToS Distribution
ToS 0   0.100
ToS 1   0.301
ToS 2   0.599
Package Sizes Distribution
500.0 750.0 1000.0 1250.0 1500.0        1.0
Package Probabilities Distribution
0.53, 0.16, 0.07, 0.1, 0.14     0.20072707637441295
0.05, 0.28, 0.25, 0.27, 0.15    0.20041816309616495
0.22, 0.05, 0.06, 0.62, 0.05    0.1999081213217135
0.1, 0.16, 0.36, 0.24, 0.14     0.19973859574218716
0.08, 0.16, 0.35, 0.21, 0.2     0.19920804346552143
```

bandwidth hist of graphs of size all

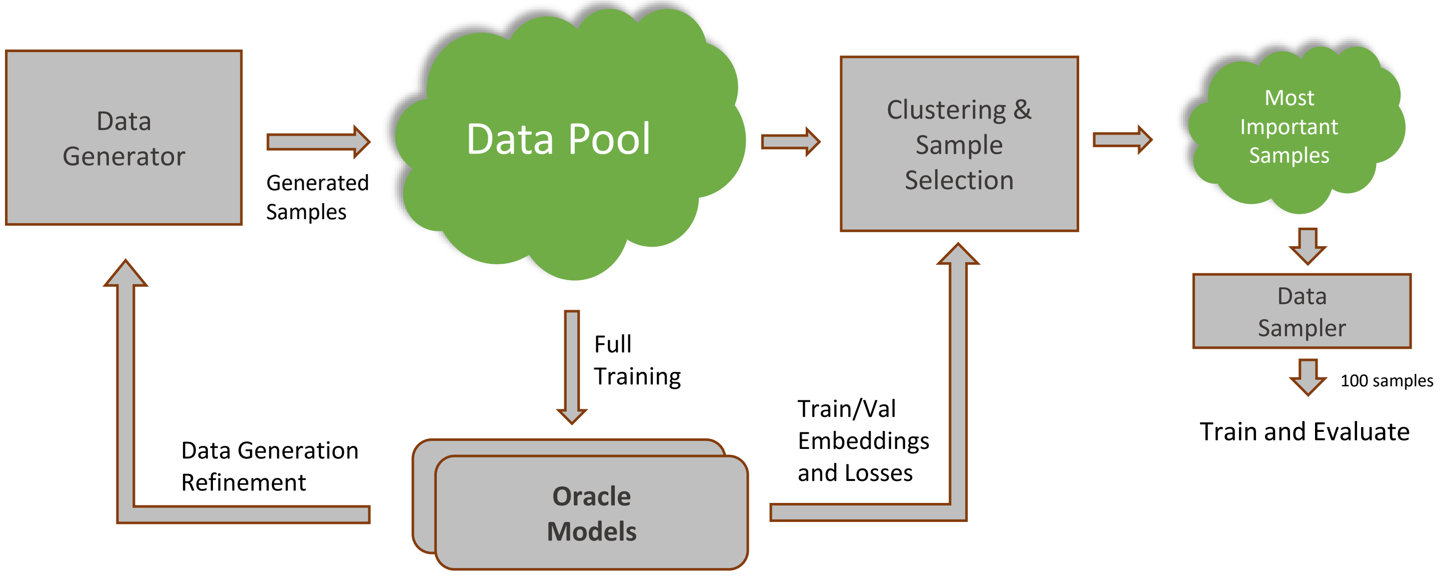


The Oracle Approach

- **Suppose we have lots of small network samples for training. How good can we get?**
  A: Challenge organizers claim MAPE < 5% on validation set
  - *"To test the capability of RouteNet-Fermi to potentially scale to larger networks, we have trained it with a very large dataset with thousands of samples of networks up to 10 nodes. After training, we could validate that the model was able to produce accurate per-path delay estimates on the validation dataset (Mean Relative Error < 5%)."*

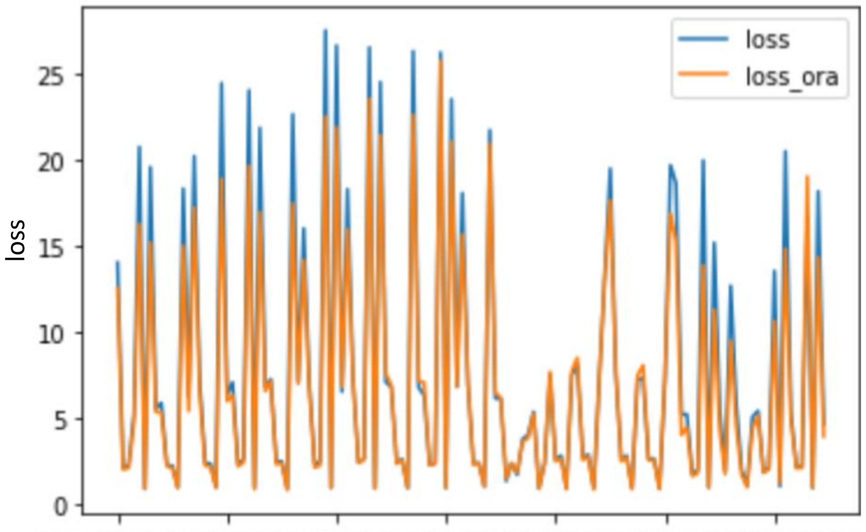
- **Can we use such a model to select a smaller set of 100 samples?**
  - Oracle model can help distinguish objectively hard examples from easy ones
  - Oracle model can be used to extract feature space embeddings for each sample

- **Approach overview**
  - Generate a large heterogenous data pool
  - Train one or more oracle models on this pool
  - Using oracle, refine the data generation process to create more useful training samples (e.g. more complex)
  - Use oracle to generate embeddings for all samples
  - Cluster the generated samples in the embedding space and select a small pool of "important" samples
  - From the small pool, sample sets of 100 for training the final model
  - Select the set of 100 with best result

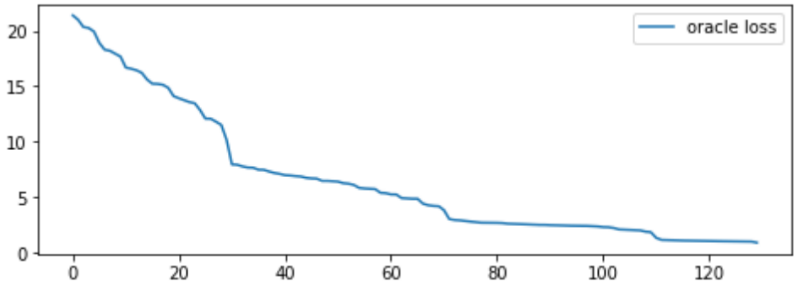


Data Analysis: Validation Set

Validation Loss
Oracle losses compared to 100-set loss

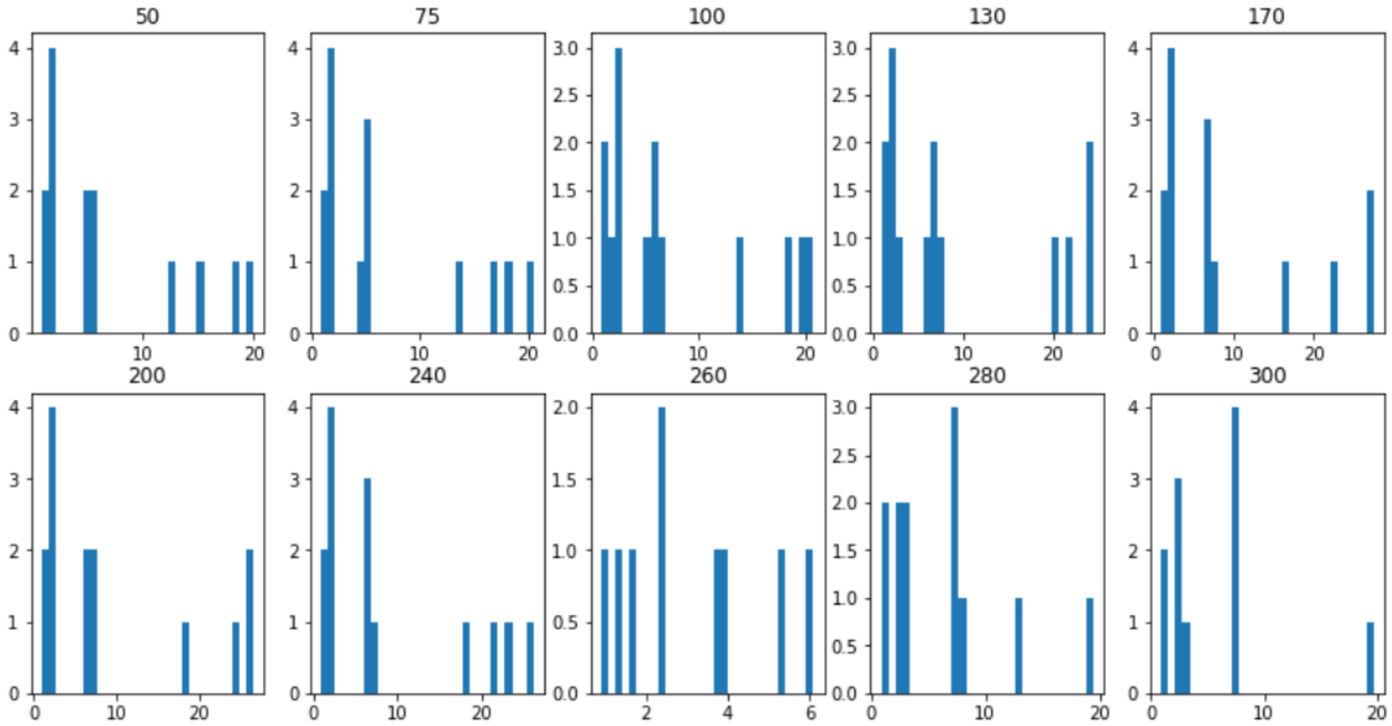


Val. Samples indices (1..130)



Val. Samples (sorted)

Oracle Loss on validation data
by graph size

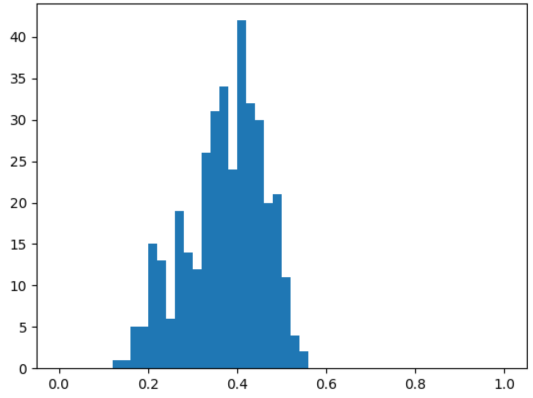


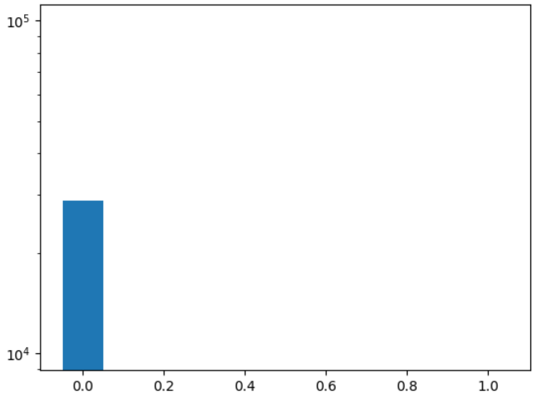
Some validation samples consistently more difficult than others. Improving on those, should improve overall loss

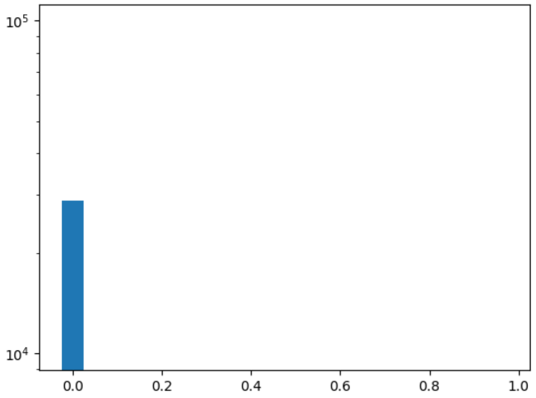
Data Analysis: Validation Set

Link Loads | Delay | Packet Drops

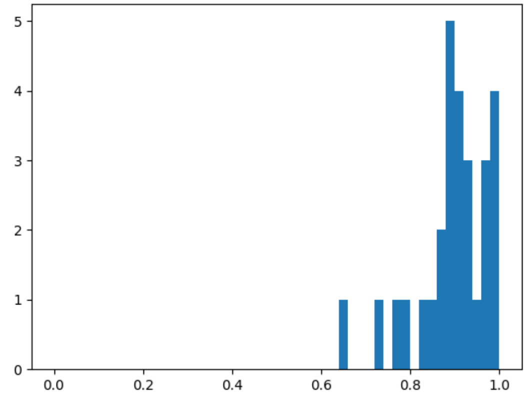
Easy Sample
170 nodes

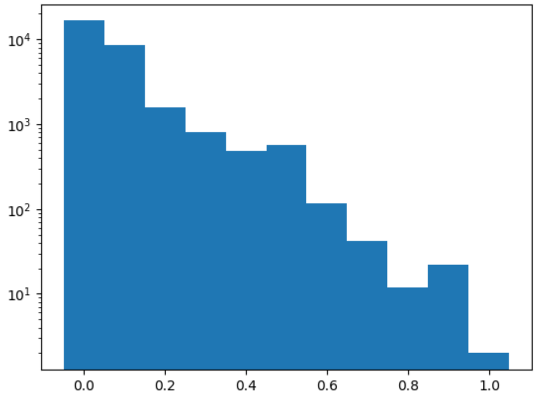


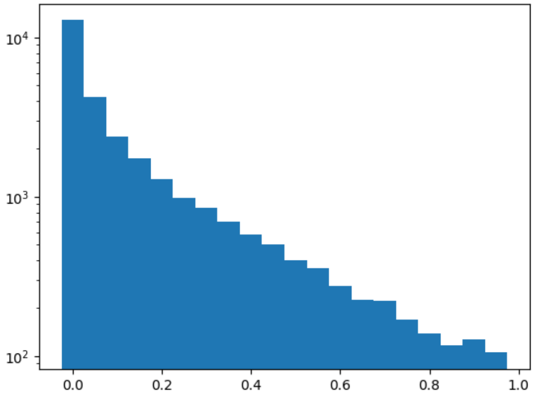




Hard Sample
170 nodes

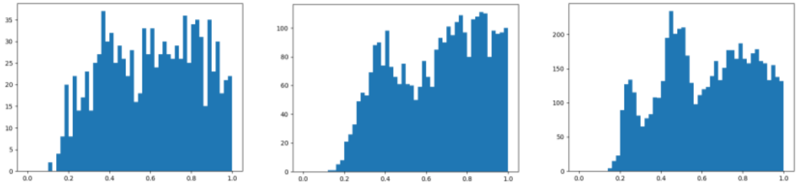






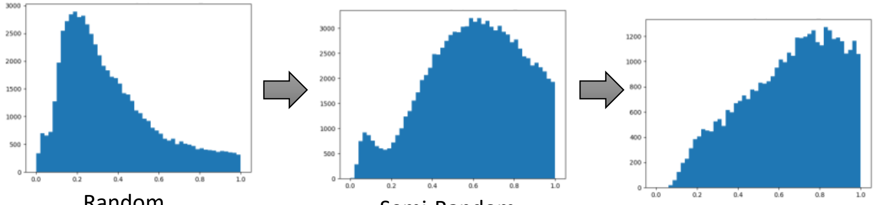
Data Analysis: Distribution Matching

Validation samples link loads



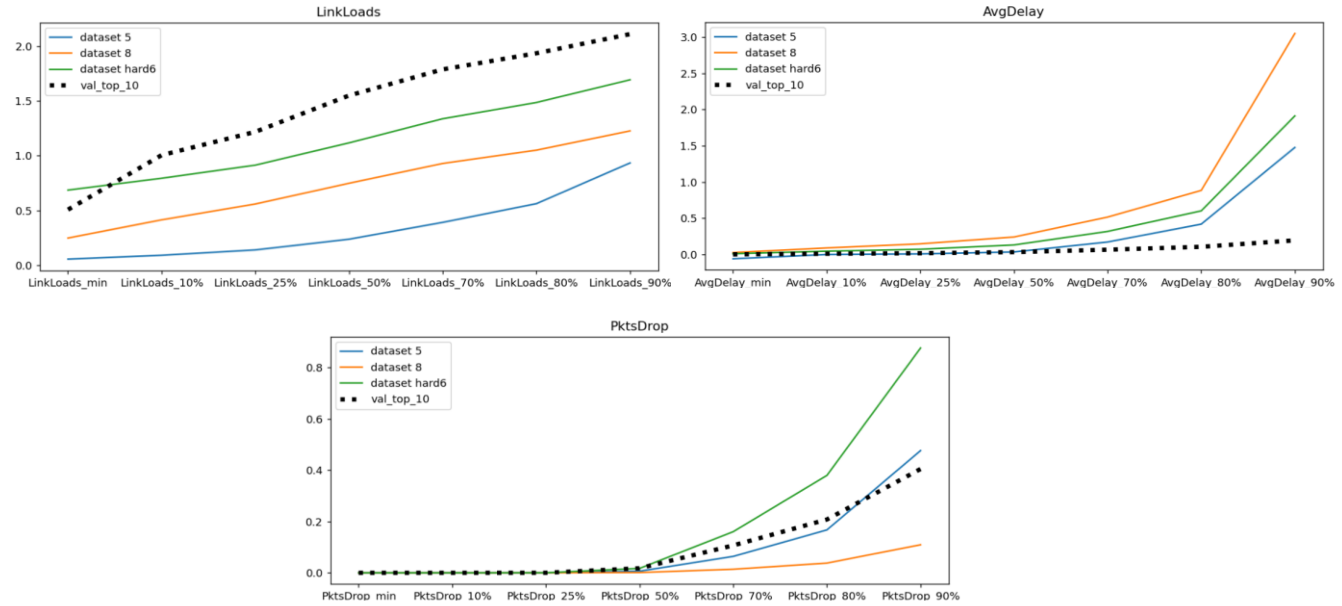
50 nodes | 170 nodes | 300 nodes

Generated samples link loads evolution



Random → Semi-Random → Hard

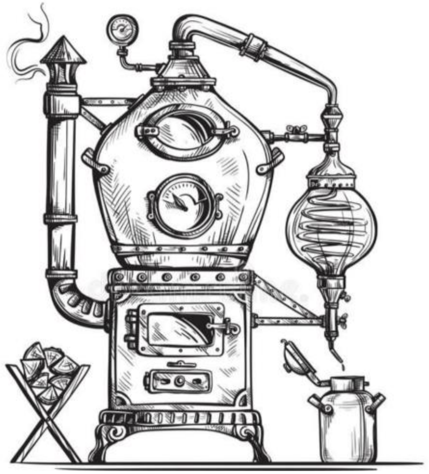
Percentile Graphs for comparing sample statistics



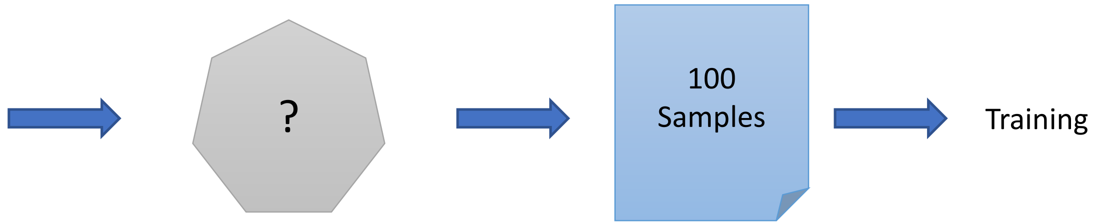
Improving the distribution of the generated data leads to better fit of the oracle model to the validation data

Data Distribution	Oracle Val. MAPE
Fully Random	7.3
Semi-Random • Mimic Validation set distributions • Random Paths	6.62
"Hard" datasets: • Link Capacities tightly match demand • Higher traffic bandwidths	5.89

Sample Selection – Select 100 samples out of 270,000 pool



- Oracle Model
- Data Pool
- Validation set



Many approaches possible for importance sampling
with or without an oracle

Select according to sample statistics

- Construct vector representation from sample KPI features statistics (e.g. Link Loads, Pkt Drops), select from K-NN of the hardest validation samples

Select according to oracle loss

- Percentile ranges

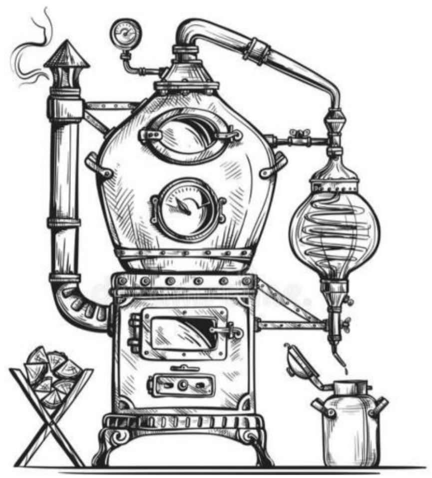
Select using oracle sample embeddings

- Apply KNN in embedding space to find nearest neighbors for each validation sample. Select from resulting pool (or use only hard validation samples)
- Cluster generated samples according to nearest validation sample -> filter clusters by distance from seed -> select from remaining groups
- K-Means cluster validation embeddings -> assign gen samples to clusters -> select near/far cluster centers

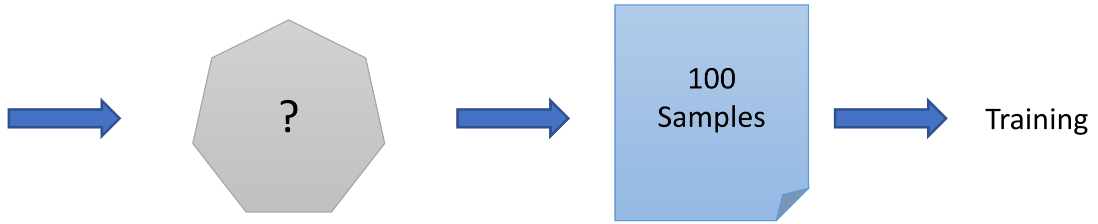
Incrementally improve a "good" set of 100

- "Evolutionary" - remove some samples, replace with randomly selected from global pool, retrain. Keep new set if good result. Repeat.
- Remove lowest loss samples from set, add nearest neighbors (in embedding space) of the hardest validation samples.
- "Friends" – for hardest samples from the set, add samples with same topology but different traffic

Sample Selection – Select 100 samples out of 270,000 pool



- Oracle Model
- Data Pool
- Validation set



Many approaches possible for importance sampling
with or without an oracle

Select using oracle sample embeddings

- Apply KNN in embedding space to find nearest neighbors for each val. sample. Select from resulting pool (or use only hard val. samples)
- Cluster generated samples according to nearest validation sample -> filter clusters by distance from seed -> select from remaining groups
- K-Means cluster val. embeddings -> assign gen samples to clusters -> select near/far cluster centers

Action Steps

- Define method for generating standard embedding for each sample
- Define cluster centers
- Distance measure to compare samples
- Define sampling strategy

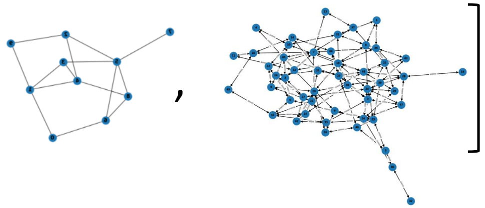
Sample Embeddings

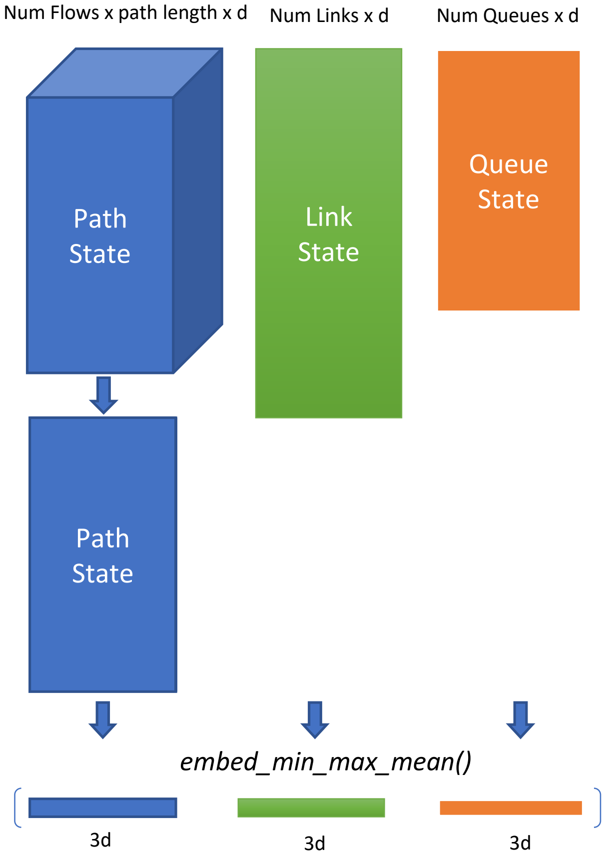
How to compare heterogenous data instances? Which metric to use?

Samples differ in many aspects - number of nodes, links, flows, queues, route path lengths and traffic distributions.

- For each data sample we create an “embedding” - a fixed size vector representation of the sample.
- Such embeddings are used for clustering and distance measurements
- We use the feature tensors encoded by the oracle model:
  - Path state: encoding of route path taken by packets along a single flow
  - Link state: encoding of links in the network, - their load and other factors
  - Queue state: encoding of queues in the network – their buffer occupancy etc.
- Tensor pooling using feature statistics – e.g. min, max, mean
- Concatenate the pooled vectors to obtain final embedding

$[G_i, R_i, T_i]$ $[G_j, R_j, T_j]$

$Dist\Big[$  $\Big] \Rightarrow Dist(e_i, e_j)$



create_sample_embeddings():

$PS \in \mathbb{R}^{F \times maxlen \times d}$: *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,* $n_f - f'th$ *item in n*

F: *number of flows in a sample*

L: *number of links in a sample*

Q: *number of queues in a sample*

$maxlen$: *maximum number of RNN steps in a flow*

d: *embedding size (32 in given code)*

 # aggregate flow features along the flow path dimension

$$\forall f: 1..F, p_f \leftarrow \frac{1}{n_f} \sum_{j=1}^{n_f} PS_{f,j,:}$$

$PS \leftarrow stack(p_f, dim = 0)$ *# this creates tensor* $PS \in \mathbb{R}^{F \times d}$

$PS_{emb} \leftarrow \text{embed\_min\_max\_mean}(PS)$

$LS_{emb} \leftarrow \text{embed\_min\_max\_mean}(LS)$

$QS_{emb} \leftarrow \text{embed\_min\_max\_mean}(QS)$

$e \leftarrow concat([PS_{emb}, LS_{emb}, QS_{emb}])$ *# resulting vector* $e \in \mathbb{R}^{9d}$

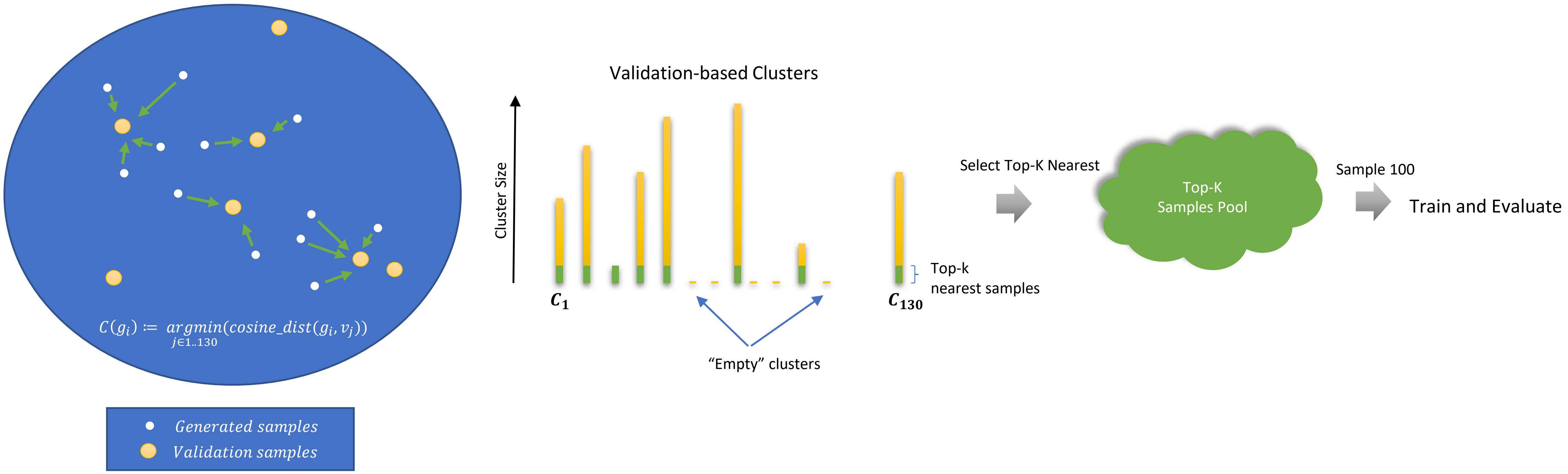
return e

 # we make use of the following helper function

function embed_min_max_mean(X):

$\quad return\ concat([\min(X, dim = 0), \max(X, dim = 0), mean(X, dim = 0)])$

Sample Selection



How to reduce the pool size?

Approach: Use samples that are "close" to the validation samples in the embedding space.

- Each cluster corresponds to a particular validation sample (130 in total)
- Assign each generated sample to a cluster with the minimum cosine distance (some have no assignment)
- Pick Top-K samples from each cluster, nearest to the cluster center (K=3, or K=5) -> this results in up to 130 groups of K samples.
- Sample sets of 100 from the resulting groups

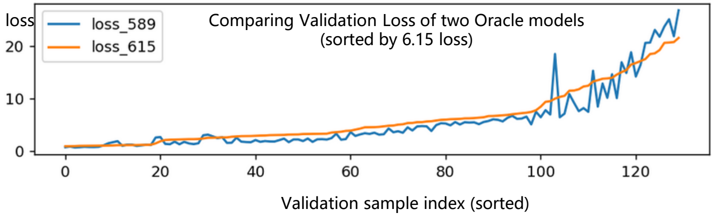
Reduce the sample pool from 270,000 to less than 500!

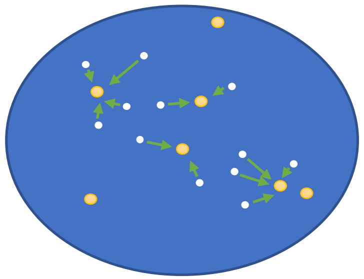
Sample Selection: Multiple Oracles

Can we do better?

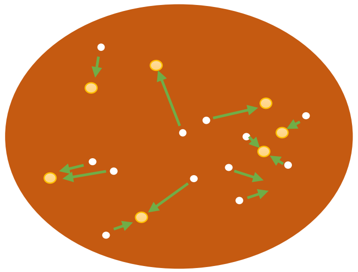
Approach: Combine embeddings from multiple oracle models

- Embeddings are not ideal
- Oracle models trained on different data distributions, perform different on the highest-loss samples
- For every cluster, use embeddings of the oracle that performs best on the clusters` validation sample





Oracle 1



Oracle 2

Method	Oracle result	Best result 100 samples
Random	N/A	8.56
Random (on selected distributions)	N/A	7.05
Single Oracle	6.44	6.69
Single Oracle	6.33	6.61
Single Oracle	6.15	6.58
Single Oracle	5.89	6.89
Two Oracles	**6.15, 5.89**	**6.31** WINNER

 # cluster assignment
<u>assign_clusters()</u>

Inputs:

n_{ora}: number of oracle models
$GenEmb^{(1)},..,GenEmb^{(n_{ora})} \in \mathbb{R}^{G\times d}$: *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*
k: *number of samples to keep in each cluster*
V: *number of validation samples*, G: *number of generated samples*, d: *embedding size*

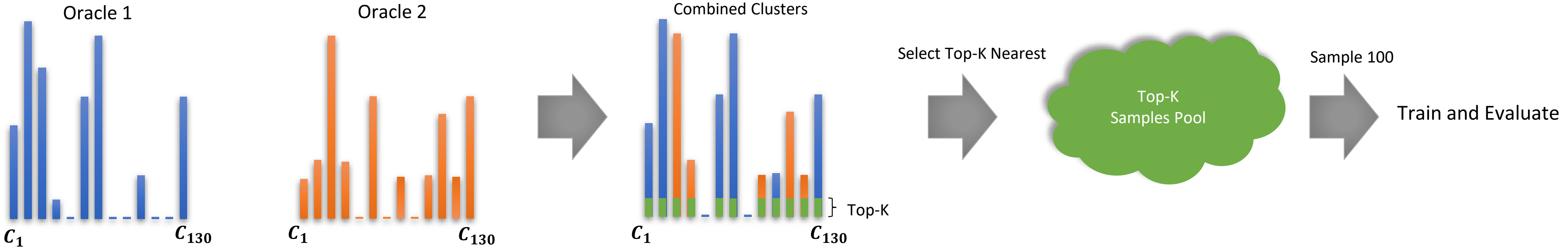
$D \leftarrow [\mathbf{0}]_{G\times V}$
$for\ v\ in\ 1..V:$
  # select oracle that performs best on given validation sample
  $\mathrm{ora} \leftarrow \underset{i=1..n_{ora}}{\mathrm{argmin}}(\mathrm{ValLoss}^{(i)}[v])$

  # compute similarity using selected oracle embeddings
  $for\ g\ in\ 1..G:$
    $D_{g,v} \leftarrow CosineDistance\left(GenEmb^{(ora)}[g], ValEmb^{(ora)}[v]\right)$

$for\ v\ in\ 1..V:$
  $CLS_v \leftarrow \left\{g \in 1..G :\ \underset{i}{\mathrm{argmin}}(D[g,i]) = v\right\}$

 # compute top − k samples in each cluster that are nearest to the cluster val. sample
$For\ v\ in\ 1..V:$
  $Sorted_v \leftarrow sort\ all\ g \in CLS_v\ in\ increasing\ order\ of\ D_{g,v}$
  $Top_v \leftarrow Sorted_v[1..k]$

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



Challenge Submissions

Final Teams Ranking

RANK	TEAM	MAPE (%)
1st	snowyowl	8.55334
2nd	Ghost Ducks	8.55446
3rd	net	9.79016
4th	iDMG	10.21327
5th	SuperDrAI	11.49168
6th	NUDTXD	11.69844
7th	FNeters	12.70369
8th	NetRevolution	13.12746
9th	GoGoGo	15.37968
10th	chainnet	18.12810
11th	Graph-X	18.66807
12th	CCI	21.18417
13th	BJIP	37.50674

Our Submissions

	Val MAPE	Test MAPE
1	7.08	9.62
2	6.71	8.99
3	7.16	9.24
4	6.68	8.89
5	6.70	8.84
6	6.96	9.01
7	6.61	8.85
8	6.65	8.78
9	6.67	9.31
10	6.89	9.16
11	6.53	8.70
12	6.78	8.91
13	6.67	9.04
14	7.02	9.08
15	6.58	9.19
16	6.54	8.73
17	6.32	8.55

Data improvements

Clustering improvements

Data improvements

2 oracles

Evaluation Stage: Oct 3 – Oct 18
Submission Budget: 20 Submissions, up to 5 per day

Questions?