

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



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH



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

It enables to reproduce the network behavior for analyzing what-if scenarios:

- What happens if change in configuration?
- What is the best upgrade given limited budget?

ML-based Digital Twin

A good training dataset is crucial for ML solutions to achieving good performance.

A good dataset requires:

- Good coverage of possible cases (e.g. congestion levels)
- Edge cases (e.g. different types of failures)
- Compact → to save training times and costs

How to produce a good dataset?

Training on data from real networks is not feasible

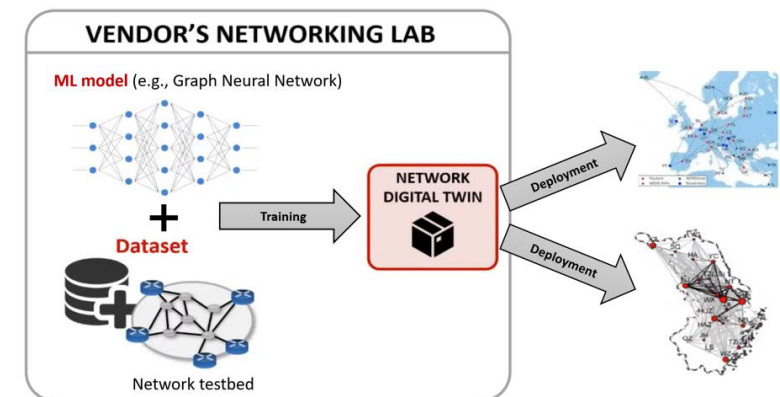
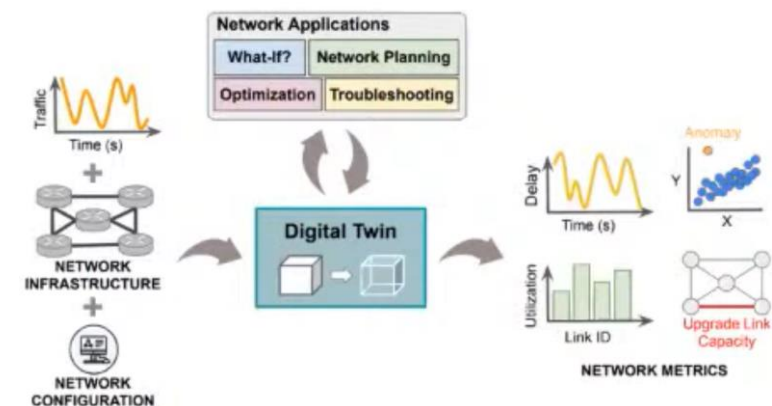
- Edge cases are rare
- High-resolution telemetry collection is costly and slow
- Privacy concerns may prohibit access to historical records

Simulating realistic traffic is not feasible, because simulation tools are very slow

There is no research on how to produce good datasets for ML models applied to networking!

Potential benefits:

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

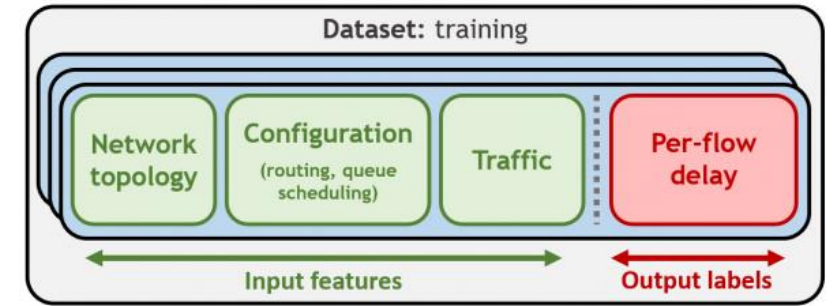


ML based Digital Twin

Problem Statement

The Goal: Produce a training dataset that scales effectively to samples of large networks.

- Participants are given
 - Packet-level network simulator to generate datasets.
 - State-of-the-art GNN model for network performance evaluation (RouteNet-Fermi)
- Dataset Generation Constraints
 - Up to 10 nodes in the network
 - Up to 100 samples
 - Fixed training regime and model architecture



Simulation Parameters

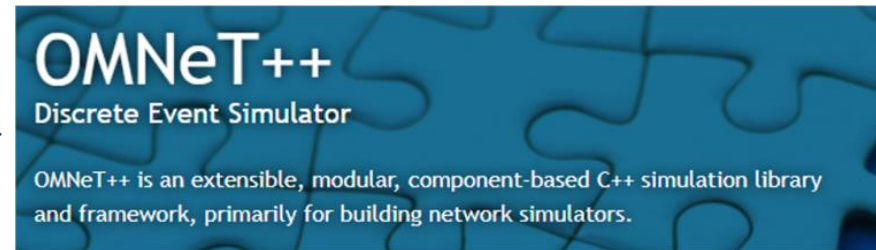
1. Topology

- 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

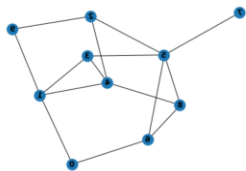
- Avg. bandwidth
- TOS – type of service (QoS)
- Packet time distribution
- Packet size distribution



Simulation Output

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

Computationally very expensive to simulate large networks

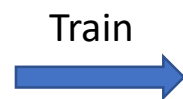


Small networks

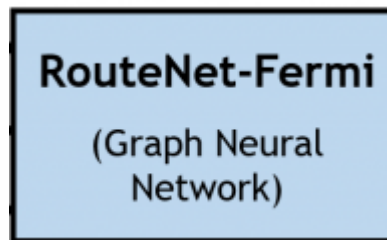
- Up to 10 nodes
- Max 90 flows
- Tight Budget**
 - Up to 100 samples



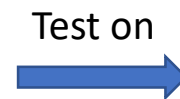
Train set
(to find)



Train



Fixed training regime



Test on

Large Networks

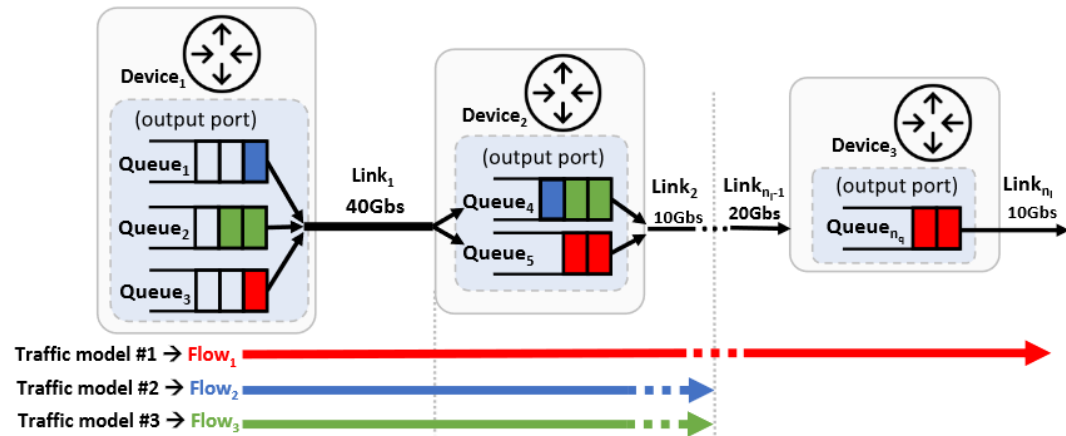
- Up to 300 nodes
- Max 89700 flows

Val and Test sets
(given)



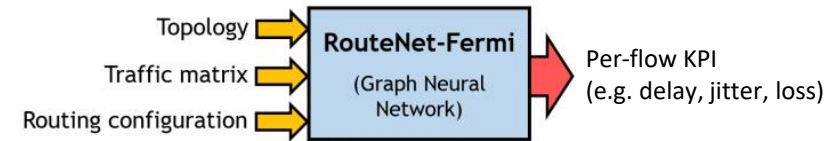
RouteNet

- A Message Passing GNN
- Input: Network configuration, Traffic
- Output: per-flow KPI
- 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



- Scales to larger graphs, via normalizing by link capacities

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



Algorithm 1 Internal architecture of RouteNet-F.

Input: $\mathcal{F}, \mathcal{Q}, \mathcal{L}, x_f, x_q, x_l$

Output: \hat{y}_{fd}

```

1: for each  $f \in \mathcal{F}$  do  $h_f^0 \leftarrow HS_f(x_f)$ 
2: for each  $q \in \mathcal{Q}$  do  $h_q^0 \leftarrow HS_q(x_q)$ 
3: for each  $l \in \mathcal{L}$  do  $h_l^0 \leftarrow HS_l(x_l)$ 

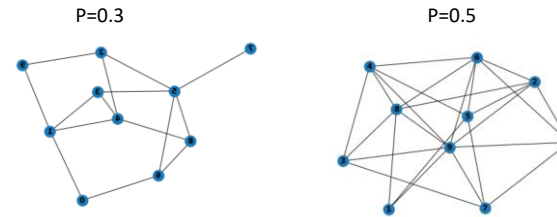
4: for  $t = 0$  to  $T-1$  do
5:   for each  $f \in \mathcal{F}$  do
6:      $\Theta([\cdot, \cdot]) \leftarrow FRNN(h_f^t, [\cdot, \cdot])$ 
7:     for each  $(q, l) \in f$  do
8:        $h_{f,l}^t \leftarrow \Theta([h_q^t, h_l^t])$ 
9:        $\tilde{m}_{f,q}^{t+1} \leftarrow h_{f,l}^t$ 
10:     $h_f^{t+1} \leftarrow h_f^t$ 
11:   for each  $q \in \mathcal{Q}$  do
12:      $M_q^{t+1} \leftarrow \sum_{f \in Q_f(q)} \tilde{m}_{f,q}^{t+1}$ 
13:      $h_q^{t+1} \leftarrow U_q(h_q^t, M_q^{t+1})$ 
14:      $\tilde{m}_q^{t+1} \leftarrow h_q^{t+1}$ 
15:   for each  $l \in \mathcal{L}$  do
16:      $\Psi(\cdot) \leftarrow LRNN(h_l^t, \cdot)$ 
17:     for each  $q \in L_q(l)$  do
18:        $h_l^t \leftarrow \Psi(\tilde{m}_q^{t+1})$ 
19:      $h_l^{t+1} \leftarrow h_l^t$ 
20:   for each  $f \in \mathcal{F}$  do
21:      $\hat{y}_{fd} = 0$ 
22:     for each  $(q, l) \in f$  do
23:        $\hat{d}_q = R_{fd}(h_{f,l}^T) / x_{l_c}$ 
24:        $\hat{d}_t = x_{f_{ps}} / x_{l_c}$ 
25:        $\hat{d}_{link} = \hat{d}_q + \hat{d}_t$ 
26:      $\hat{y}_{fd} = \hat{y}_{fd} + \hat{d}_{link}$ 

```


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



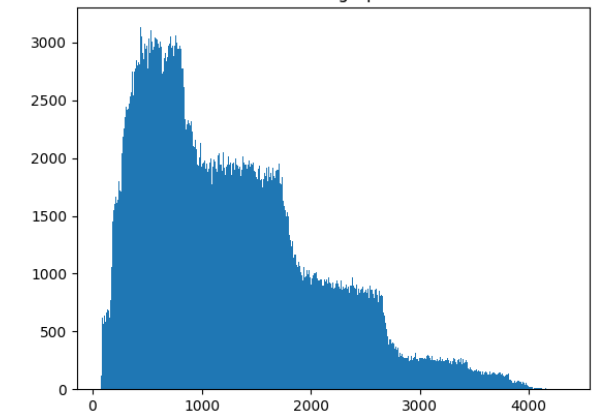
Val. Sample - 50 nodes



```

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?**

Challenge organizers claim $\text{MAPE} < 5\%$ on validation set

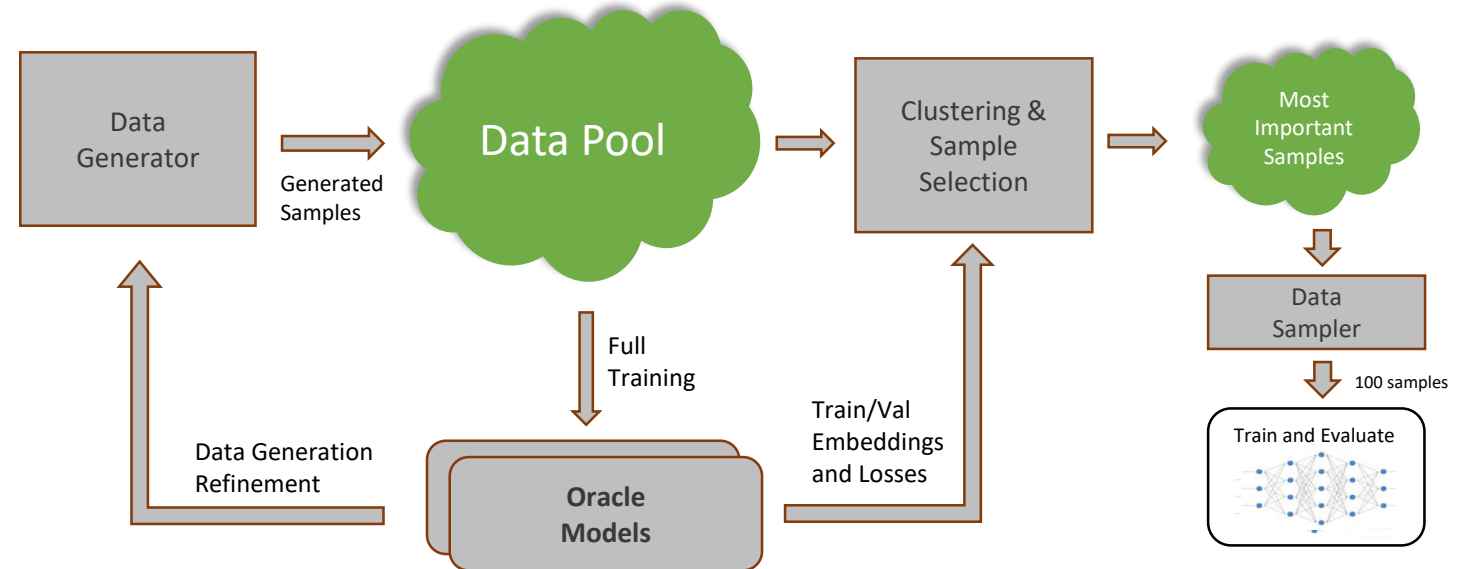
- *“To test the capability of RouteNet-Fermi to potentially scale to larger networks, we have trained it with a very large dataset with thousands of samples of networks up to 10 nodes. After training, we could validate that the model was able to produce accurate per-path delay estimates on the validation 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

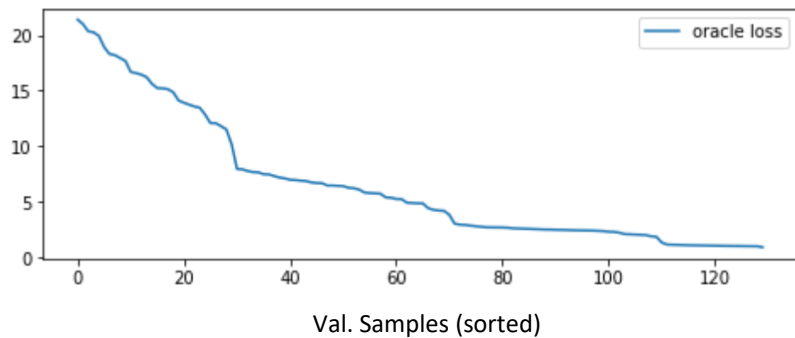
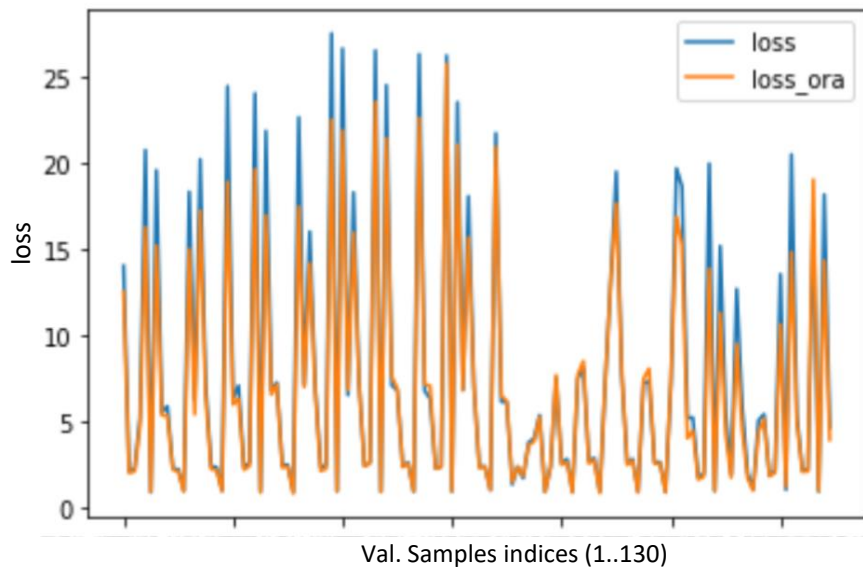
- **Approach overview**

- Generate a large heterogenous data pool
- Train one or more oracle models on this pool
- Using oracle, refine the data generation process to create better training samples
- 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

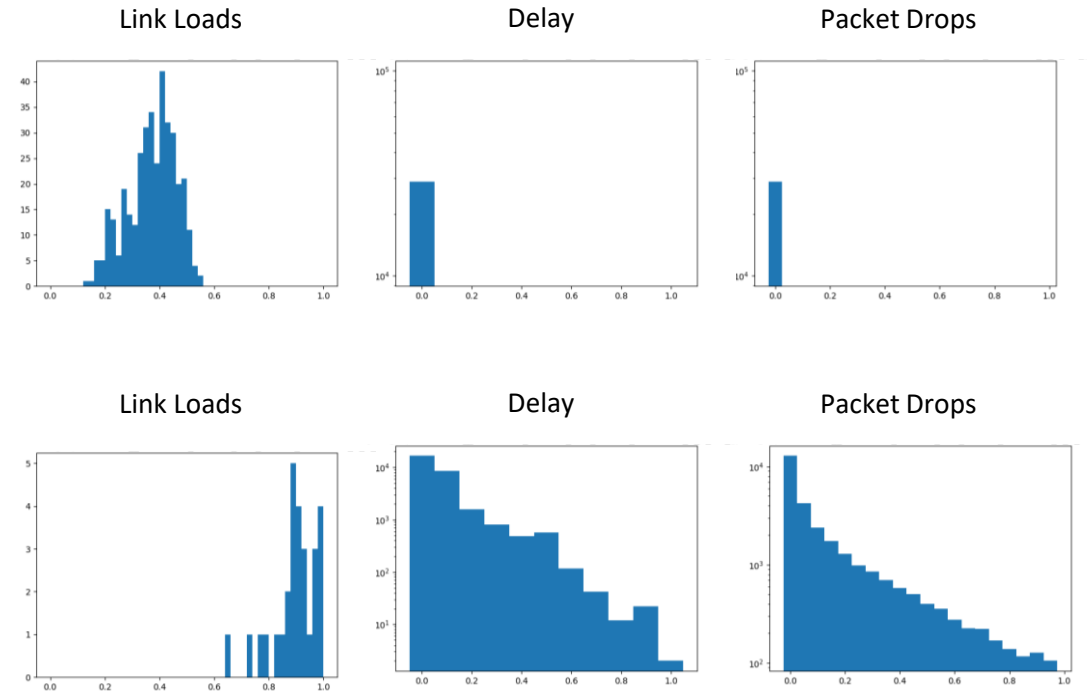


Validation Set Analysis

Validation Loss
Oracle losses compared to 100-set loss



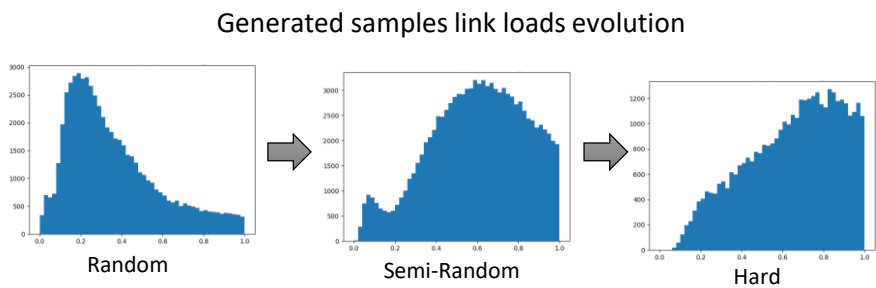
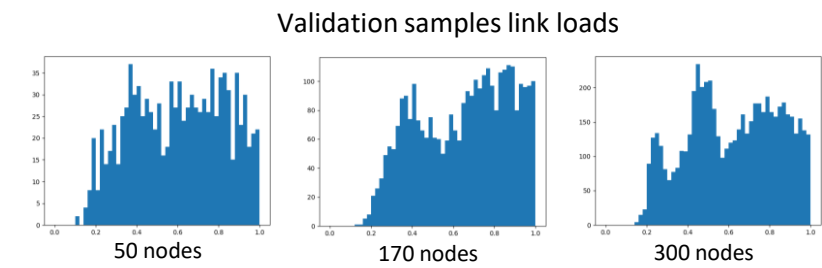
Easy Sample
170 nodes



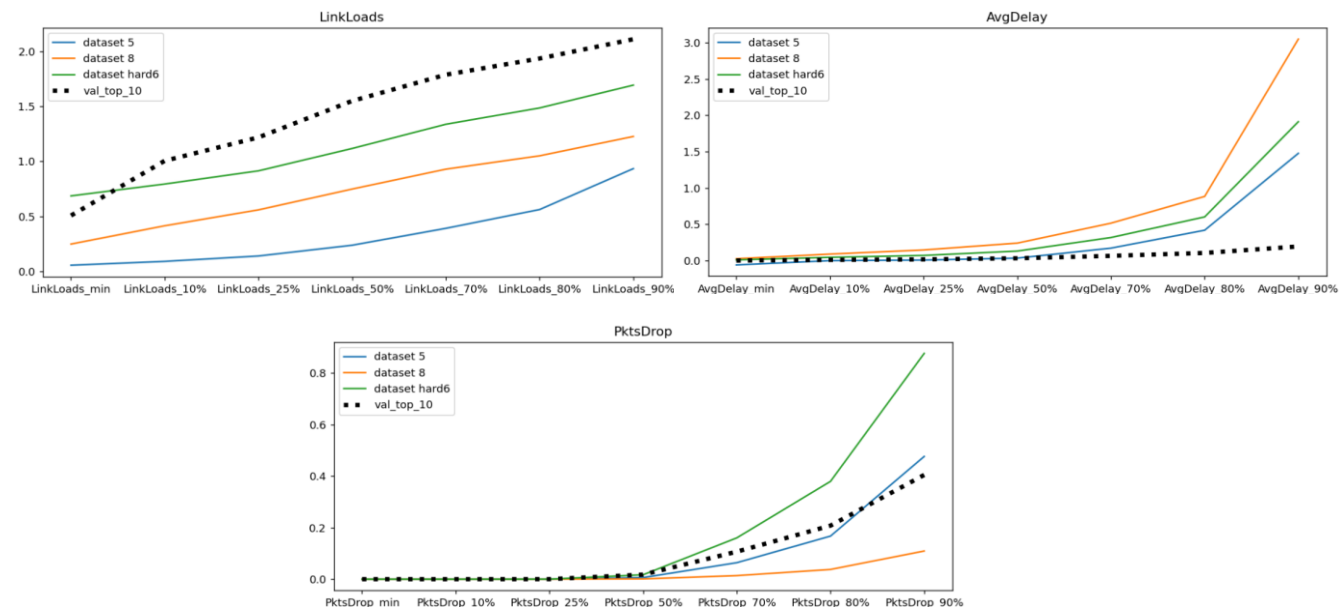
Hard Sample
170 nodes

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

Data Analysis: Distribution Matching



Percentile Graphs for comparing sample statistics

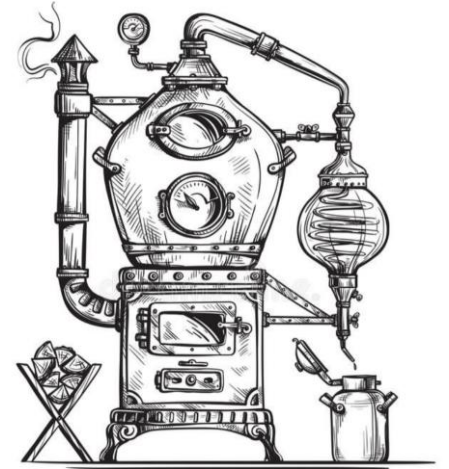
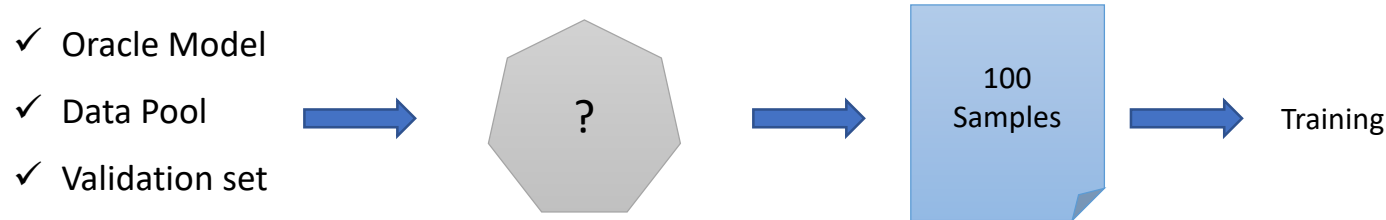


Oracle Training

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 <ul style="list-style-type: none">Mimic Validation set distributionsRandom Paths	6.62
“Hard” datasets: <ul style="list-style-type: none">Link Capacities tightly match demandHigher traffic bandwidths	5.89

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



Many approaches possible for importance sampling

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

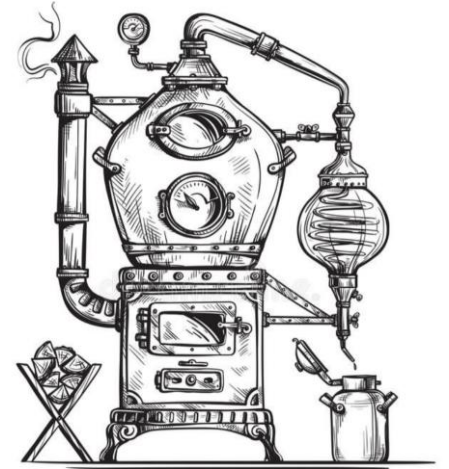
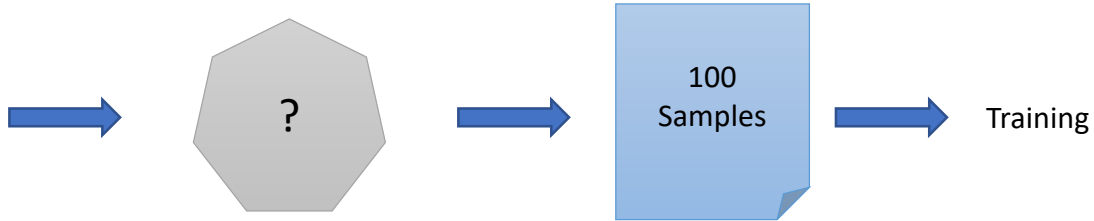
- Apply KNN in embedding space to find nearest neighbors for each validation sample → Select from resulting pool (or use only hard validation samples)
- Cluster generated samples according to nearest validation sample → Filter clusters by distance from seed → Select from remaining groups
- K-Means cluster 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



Select using oracle sample embeddings

- ★ • Cluster generated samples according to nearest validation sample → Filter clusters by distance from seed → Select from remaining groups

Action Steps

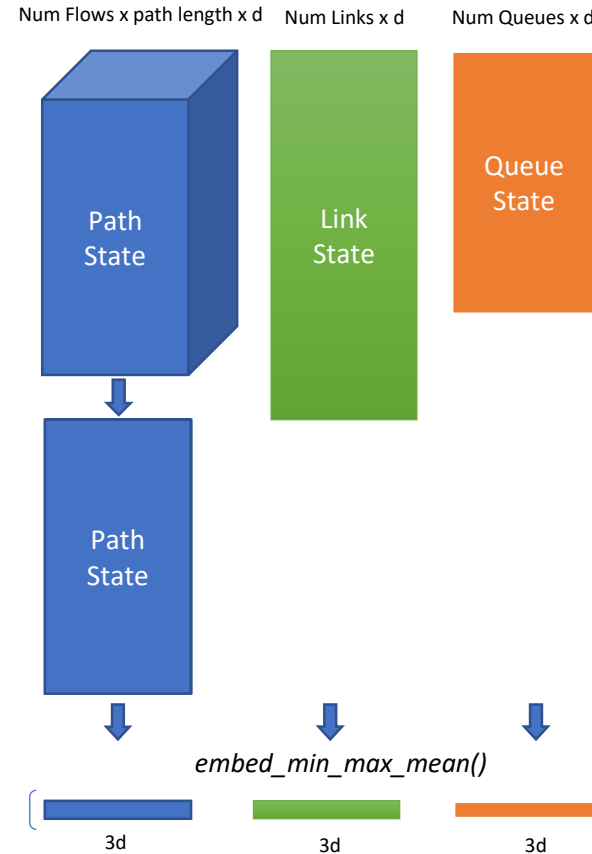
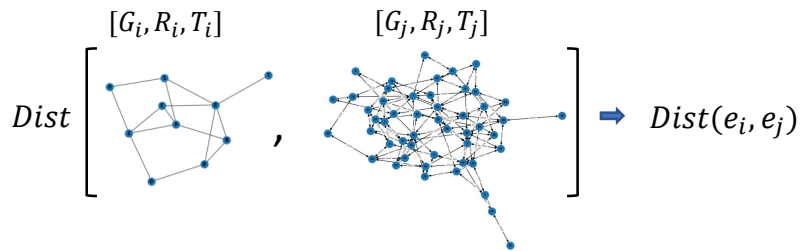
- Define method for generating standard embedding for each sample
- Define cluster centers
- Distance measure to compare samples
- Define sampling strategy

Sample Embeddings

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

Samples differ: number of nodes, links, flows, queues, route path lengths and traffic distributions

- Create “embedding” for each data sample - a fixed size vector representation
- Use embeddings for clustering and distance measurements
- Embedding is composed of oracle model feature tensors:
 - **Path state**: encodes route paths taken by packets along a single flow
 - **Link state**: encodes network links - load and other factors
 - **Queue state**: encodes network queues – buffer occupancy etc.
- Tensors are pooled using feature statistics – e.g. min, max, mean



```

create_sample_embeddings():
PS ∈ ℝF×maxlen×d: path state tensor
LS ∈ ℝL×d: link state tensor
QS ∈ ℝQ×d: queue state tensor
n ∈ ℝF: vector containing number of steps in every flow path, nf – 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

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

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

$PS_{emb} \leftarrow \text{embed_min_max_mean}(PS)$

$LS_{emb} \leftarrow \text{embed_min_max_mean}(LS)$

$QS_{emb} \leftarrow \text{embed_min_max_mean}(QS)$

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

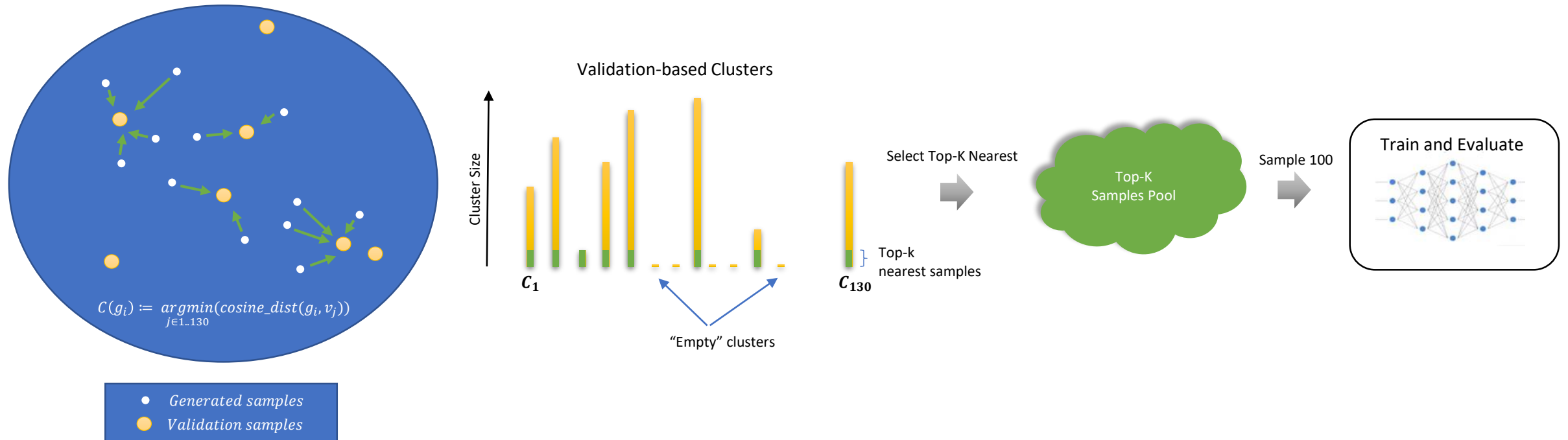
return e

we make use of the following helper function

```

function embed_min_max_mean(X):
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.

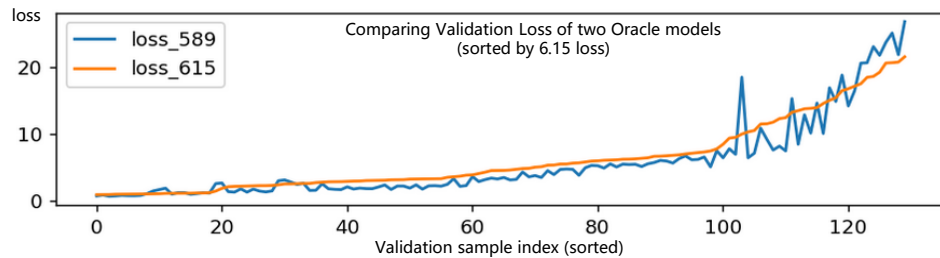
- Each cluster corresponds to a particular validation sample (130 in total)
- Assign each generated sample to a cluster with the minimum cosine distance
- Pick Top-K samples from each cluster, nearest to the cluster center (K=3, or K=5) → this results in up to 130 groups of K samples
- Sample sets of 100 from the resulting groups

Cluster Selection reduced the sample pool from 270,000 to less than 500!

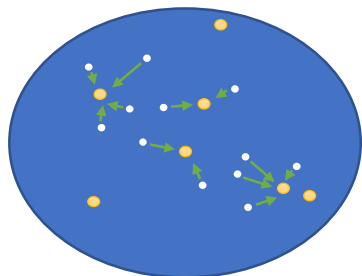
Sample Selection: Multiple Oracles

Can we do better?

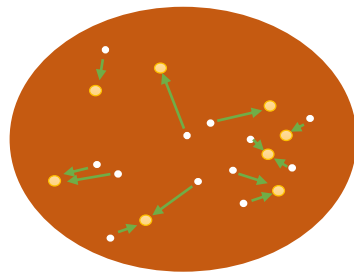
Yes - Combine embeddings from multiple oracle models



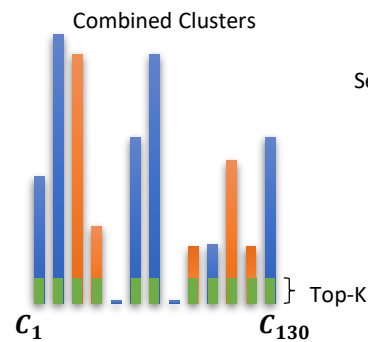
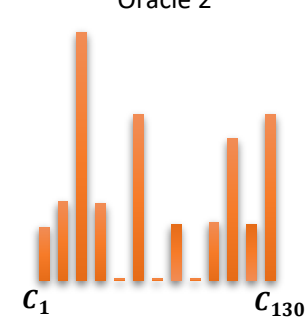
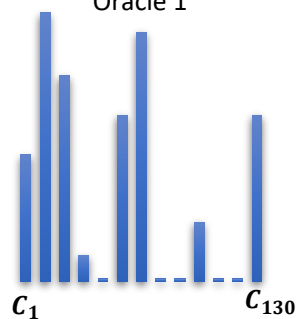
- Oracle models trained on different data distributions, have different strengths and weaknesses
- Embeddings are not ideal
- Select embeddings of the best oracle



Oracle 1



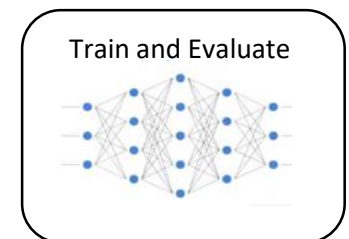
Oracle 2



Select Top-K Nearest



Sample 100



```
# cluster assignment
assign_clusters()

Inputs:
nora: number of oracle models
GenEmb(1), ..., GenEmb(nora) ∈ ℝG×d : embeddings of generated samples from different oracles
ValEmb(1), ..., ValEmb(nora) ∈ ℝV×d : embeddings of the validation samples from different oracles
ValLoss(1), ..., ValLoss(nora) ∈ ℝV : 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 ← [0]G×V
for v in 1..V:
    # select oracle that performs best on given validation sample
    ora ← argmini=1..nora(ValLoss(i)[v])

    # compute similarity using selected oracle embeddings
    for g in 1..G:
        Dg,v ← CosineDistance(GenEmb(ora)[g], ValEmb(ora)[v])

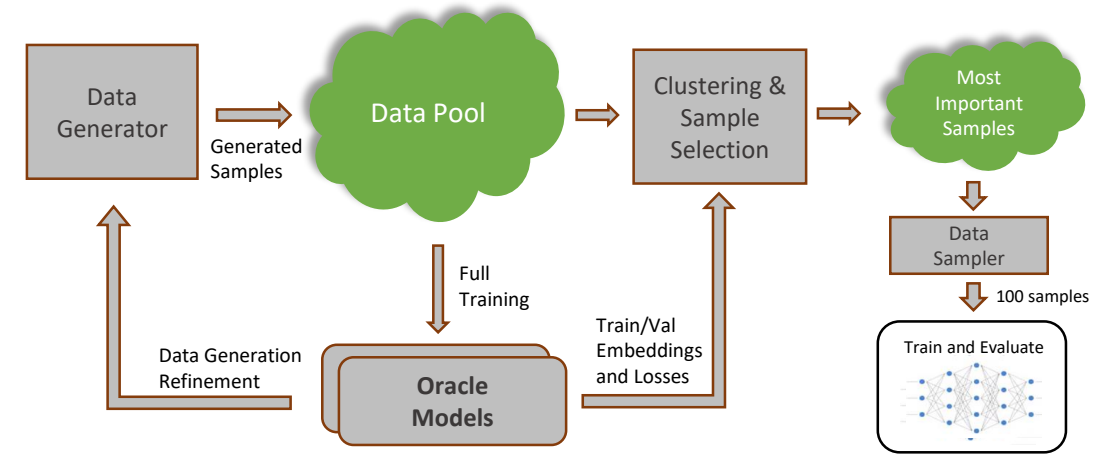
for v in 1..V:
    CLSv ← {g ∈ 1..G : argmini(D[g, i]) = v}

# compute top-k samples in each cluster that are nearest to the cluster val. sample
For v in 1..V:
    Sortedv ← sort all g ∈ CLSv in increasing order of Dg,v
    Topv ← Sortedv[1..k]


return {Topv for v in 1..V}
```

Results and Summary

- Generate a large heterogenous data pool
- Combine several Oracle models to project the generated samples into optimal embedding space
- 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



Methods Comparison
(validation MAPE)

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 

Our Submissions

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

Final Ranking

RANK	TEAM	MAPE (%)
1 st	snowyowl	8.55334
2 nd	Ghost Ducks	8.55446
3 rd	net	9.79016

RANK	TEAM	MAPE (%)
1 st	snowyowl	8.55334
2 nd	Ghost Ducks	8.55446
3 rd	net	9.79016
4 th	iDMG	10.21327
5 th	SuperDrAI	11.49168
6 th	NUDTXD	11.69844
7 th	FNeters	12.70369
8 th	NetRevolution	13.12746
9 th	GoGoGo	15.37968
10 th	chainnet	18.12810
11 th	Graph-X	18.66807
12 th	CCI	21.18417
13 th	B3IP	37.50674

Further Improvements – after competition has ended

Oracle training speed-up

- **Current Approach:**
 - Domain expert data generation
 - Manual selection of a refined sub dataset
 - Train oracle
- **Rapid Training**
Incremental reduction of Data Pool, based on sample embeddings
 - *Init*: Train-dataset defined to the whole ‘Data pool’ of 270K samples
 - *Step 1*: Train oracle on train-dataset, *with early stopping*
 - *Step 2*: Select smaller train-dataset using oracle-based clustering
 - Return to *Step 1*

No need for manual time-consuming data selection
Less than 1 epoch to get to a good oracle model using early stopping

Data Distribution	Oracle Val. MAPE	Single Oracle (100 samples)
“Hard” datasets	5.89	6.89

Data Distribution	Oracle Val. MAPE	Early stop	Single Oracle (100 samples)
Full pool (270K)	6.42	1 epoch	-
Refined pool (6500K)	5.89	20 epochs	6.67

Oracle embeddings improvement

- **Current Approach:** *min max mean*
- **Improved:** *Added quantiles data to pooled feature tensor statistics*

Method	Oracle Result	100 samples Original Emb.	100 samples Quantile Emb.
Single Oracle	6.15	6.58	6.48
Single Oracle	5.89	6.89	6.61
Two Oracles	6.15, 5.89	6.31	6.30



Questions?