Solution Outline

Our solution consists of the following high-level steps.

- Generating a large pool of samples that fit the challenge constraints
- Training the RouteNet architecture on a large dataset in order to get a well-trained model which achieves low MAPE on the validation set (similar as was described by challenge organizers). We refer to this model as “oracle”.
- Using the oracle model, extract vector representations (embeddings) for all the generated samples and cluster them according to similarity with validation data samples.
- Sample and train sets of 100 from the created clusters, select the best sets for submission.

Generating samples

Overall, during the course of the challenge we generated around 270K samples. All the samples are randomly generated. We had several iterations of data generation, each time with different distribution.

To assess the quality of generated data, we trained an oracle model to see how low MAPE we can get on the validation set. As we improved our data distribution to better fit the one in validation data, so we saw improvements both in the oracle model result and in the results of training various sets of 100 from that pool.

Initially we began with an entirely random approach where all the options for link loads, queues, traffic etc. were sampled uniformly from the range of valid values in each category (datasets 0-5). The sample graphs were generated using erdos_renyi method with edge probability ranging between 0.1 and 0.5. 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. This data distribution gave us oracle that reach around 7.3 validation MAPE.

Next, we analyzed validation set parameters – link bandwidths, buffer sizes, packet size distributions etc. – and generated data from a similar distribution (datasets 6-15). We noticed that many fields in the validation data have limited ranges of values – e.g. the on-off time distribution always has 5 sec on and off period – and we updated our configuration to have those limited ranges as well. In this stage we also modified the routing algorithm so that the routes it chooses are random path between two nodes instead of shortest path. This improved the oracle model to around 6.6 MAPE.

Finally, we tried to match the KPI of the generated samples – link loads, packet drop ratio and delay to those in the validation data (datasets hard1-hard6). To this end, we increased average bandwidth in the samples and also increased the link capacities to reduce delays. We also generated dataset where the link capacity is not randomly sampled but is derived from the total average bandwidth of the flows that pass through a link. In such data, the link capacity is set either to the nearest value or nearest value above the total bandwidth, from the given set [10000, 25000, 40000, 100000, 250000, 400000]. These modifications further improved the oracle. The best validation MAPE we got was 5.89.

Training the oracle model

We trained the oracle RouteNet model with the following parameters:

200 epochs, 10000 steps per epoch

Learning rate: 1e-3 with step decay every 15 epochs with factor 0.75

The two oracle models we used, reaching 6.15 and 5.89 validation MAPE, were trained on about 85K samples taken from sub-datasets where routing is random path, link loads and other values are similar to those appearing in validation data and average bandwidths are relatively high. (please see our instructions doc for how to generate all the sets and which data distributions exactly is used in training our oracle models)

Extracting Sample Embeddings

Once we have a well-trained oracle model, we use it to extract a vector representation – "embedding" – for each sample in the validation data and the training data. We modified RouteNet model code to return from forward pass the tensors - path_state, link_state, queue_state and length of each flow path. We use these to generate the embedding vectors. Below is pseudocode for generating an embedding vector for a given sample:

$\underline{create\ sample\ embeddings(\ ):}$
$PS \in \mathbb{R}^{F \times maxlen \times d}: path\ state\ tensor$
$LS \in \mathbb{R}^{L \times d}: link\ state\ tensor$
$QS \in \mathbb{R}^{Q \times d}: queue\ state\ tensor$
$n \in \mathbb{R}^{F}: vector\ containing\ number\ of\ steps\ in\ every\ flow\ path, n_f - f'th\ item\ in\ n$
$F: number\ of\ flows\ in\ a\ sample$
$L: number\ of\ links\ in\ a\ sample$
$Q: number\ of\ queues\ in\ a\ sample$
$maxlen: maximum\ number\ of\ RNN\ steps\ in\ a\ flow$
$d: embedding\ size\ (32\ in\ given\ code)$

$\#\ aggregate\ flow\ features\ along\ the\ flow\ path\ dimension$

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

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

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

$LS_{emb} \leftarrow$ embed_min_max_mean(LS)
$QS_{emb} \leftarrow$ embed_min_max_mean(QS)

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

$return\ e$

$\#\ we\ make\ use\ of\ the\ following\ helper\ function$

$\underline{function\ embed\_min\_max\_mean(X)}$:
    $return\ concat([\min(X, dim = 0), \max(X, dim = 0), mean(X, dim = 0)])$

We used two oracle models, so we generate two sets of embeddings – one for each oracle.

Clustering generated samples and sampling from the clusters

Each of our clusters corresponds to a particular validation sample.

First, for each validation sample, we determine which oracle will be used for measuring distances between it and other samples. We pick the oracle that has lowest loss on that sample and use its' embeddings. Then we compute cosine distances[1] between all generated sample embeddings and all the validation embeddings.

Each generated sample is assigned to the cluster of the validation sample with the minimum distance. This way we get cluster assignments for all samples (though some validation clusters may result in having no samples assigned to them). In our solution we get 88 non-empty clusters.

Then we take top k samples from each cluster (k=3 or 5) from cluster centers (i.e. those that have lowest distance from the cluster validation sample).

Finally, we sample from these top-k groups, to arrive at a set of 100 for training. We try to take same number of samples from every group.
Here is the outline pseudocode of clustering and sampling logic:

$\#\ cluster\ assignment$
$\underline{assign\_clusters(GenEmb^{(1)}, ValEmb^{(1)}, ValLoss^{(1)}, GenEmb^{(2)}, ValEmb^{(2)}, ValLoss^{(2)}, k)}$

$Inputs$:
$GenEmb^{(1)}, GenEmb^{(2)} \in \mathbb{R}^{G \times d} : embeddings\ of\ generated\ samples\ using\ oracle\ 1\ or\ 2$
$ValEmb^{(1)}, ValEmb^{(2)} \in \mathbb{R}^{V \times d} : embeddings\ of\ the\ validation\ samples\ using\ oracle\ 1\ or\ 2$
$ValLoss^{(1)}, ValLoss^{(2)} \in \mathbb{R}^{V} : validation\ sample\ losses\ using\ oracle\ 1\ or\ 2$
$k: number\ of\ samples\ to\ keep\ in\ each\ cluster$
$V: number\ of\ validation\ samples, G: number\ of\ generated\ samples, d: embedding\ size$

$D \leftarrow [\mathbf{0}]_{\mathrm{G \times V}}$
$for\ v\ in\ 1..V$:
  $\#\ select\ oracle\ that\ performs\ best\ on\ given\ validation\ sample$
  $if\ ValLoss^{(1)}[v] \leq ValLoss^{(2)}[v]$
    $ora \leftarrow 1$
  $else$
    $ora \leftarrow 2$

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

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

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

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

$\#\ sampling\ from\ clusters$

$\#\ we\ try\ to\ sample\ equally\ from\ each\ cluster\ until\ we\ reach\ 100\ samples$

$\underline{sample\_from\_clusters(\{Top_v\ for\ v\ in\ 1..V\})}$

$100set\ = \emptyset$
$while\ |100set| < 100$:
$remainder \leftarrow 100 - |100set|$
$s \leftarrow [random\_sample(Top_v \setminus 100set, 1)\ for\ v\ in\ 1..V]$
$if\ |s| > remainder$:
$s \leftarrow random\_sample(s, remainder)$
$100set \leftarrow 100set \cup s$

$return\ 100set$

---

[1] CosineDistance$(u, v) = 1 - \frac{u \cdot v}{\|u\|_2 \cdot \|v\|_2}$

In our solution, we used k=5 when building cluster groups. This results in 88 non-empty groups, where 23 groups use embeddings from 6.15 oracle and the remaining 65 use embeddings of 5.89 oracle. In total, from the full dataset of 270K candidate samples, our final pool contains just 401 samples. From this pool, we sampled sets of 100 as described above. Our best result set, used in the submission, got a validation score of 6.32.