Solution of Team BUPT\_CMCC

## Basic Information

Our solution is implemented as an extension of the baseline RouteNet implementation provided by BNN. Specifically, we train a model for each of the two modes CBR+MB and MB respectively. During the evaluation phase, we first classify the category of each sample and then use the corresponding model to predict the delay of that sample.

We use a 24-core Intel i9-12900KS CPU to train the models without using any GPU. For each model, we train 50 epochs through 5-fold cross-validation. The whole training process for five fouls takes about 9 hours and 6 hours respectively for the CBR+MB model and the MB model.

## Detailed Features

The dataset for this competition contains two traffic patterns, CBR+MB and MB. we found that the baseline model utilizes only three flow-level features: "flow\_traffic", "flow\_packets", and "flow\_packet\_size". We believe that this approach has the following limitations:

(1) The original model can only sense the average characteristics of the traffic and cannot sense the instantaneous traffic characteristics.

(2) The original model does not learn features about time, such as packet delivery timestamps, delivery intervals, and burst intervals.

To this end, we extract some new features for the two traffic models. For the MB model, we add **Inter-Burst-Gap (IBG)** and **on-Rate**; for the CBR+MB model, we add **Mean** **Inter-Packet-Gap Gap (IPG-Mean), Variance of** **Inter-Packet-Gap Gap (IPG-Var)** and **on-Rate**. These combinations of features are the ones that work best after we repeatedly tried various features and filtered them out.

Since the end-to-end delay should be greater than the sum of transmission delay along the path. We add the following judgments to our model inference.

queue\_delay = tf.math.reduce\_sum(queue\_delay\_sequence, axis=1)

transmission\_delay = tf.math.reduce\_sum(transmission\_delay\_sequence, axis=1)

transmission\_delay = transmission\_delay \* 1e3

condition = queue\_delay < transmission\_delay

delay = tf.where(condition, transmission\_delay, queue\_delay)

return delay

In the evaluation phase, we use two models to predict samples in the test set. We modify the **predict.py** file to implement that.

# determine the type of sample for predicting separately

if (sample\_type(sample\_features) == 'CBR+MB'):

model = cbr\_mb\_model

else:

model = mb\_model

Furthermore, we adjust various hyperparameters to optimize the model. The hyper-parameterswe finally chose are shown below.

|  |  |
| --- | --- |
| **Hyper-parameters** **of CBR+MB model** | |
| iterations | 10 |
| path\_state\_dim | 64 |
| link\_state\_dim | 64 |
| epoch | 50 |
| loss | MAPE |
| optimizer | Adam |
| Learning rate | 0.001 |

|  |  |
| --- | --- |
| **Hyper-parameters** **of MB model** | |
| iterations | 8 |
| path\_state\_dim | 64 |
| link\_state\_dim | 64 |
| epoch | 50 |
| loss | MAPE |
| optimizer | Adam |
| Learning rate | 0.001 |

We attach a **requirement.txt** to the source code files, which contains our environment dependencies. In addition, we also include a **Readme.md**, in which we describe the project structure, how to process the dataset and extract the features, how to train the model, and how to predict the test set.

**Contact email: wangzicheng@bupt.edu.cn**