Graph Neural Networking Challenge 2023: m0b1us

Cláudio Modesto[†], Rebecca Aben-Athar[†], Andrey Silva[‡], Silvia Lins[‡]

[†]*LASSE / Federal University of Pará (UFPA), Brazil*

[‡]*Ericsson Research Brazil*

I. Introduction

This report represents the overall description of our proposed solution for Graph Neural Networking 2023 using the RouteNet-Fermi model [1]. The first generic (but relevant) step of our solution is the modification of the layers parameters and hyperparameters presented in the baseline source code. Besides these modifications (hyperparameters tuning), our solution relies on three more steps. The second step corresponds to feature selection of the available input features. In this step, we performed experiments using two different approaches: the filter-based and the Wrapper-based. With these feature selection experiments, we could identify relevant features for the prediction of the mean per-flow delay. The next step, corresponds to the modification of the input features normalization algorithm, replacing the min-max normalization with the Z-score; intended to make our model more reliable in dealing with outliers present in each feature distribution. The last step is related to getting together two structures commonly used separately: the attention mechanism and the message-passing framework, in such a way as to leverage the multiple-stage message-passing structure already provided by the baseline together with an attention layer; performing a kind of fine-tuning in the selected features, assigning different weights depending on the relevance of each feature.

II. Details of Adopted Steps

A. Hyperparameters Tuning

This hyperparameter tuning step can be splitted into two sub-steps. The first one is related to general parameters that define the configuration in the training, e.g., the number of epochs, the learning rate, etc. The second one is related specifically to the RouteNet model itself, e.g., the number of iterations, embedding length, etc. Tables I and II are showing the main parameters used in our solution. Training optimization hyperparameters are related to *std_train.py* file; whereas the RouteNet hyperparameters are associated with *std_models.py* file.

Parameter	Value
Epochs	150
Learning Rate	0.001
Optimizer	AdamW
Metric	MAPE
Loss Function	MAPE

TABLE I: Training optimization Hyperparameters

Parameter	Value
Iterations	12
Link Embedding Length	16
Flow Embedding Length	16
LeakyReLU Slope(α)	0.01

TABLE II: RouteNet optimization Hyperparameters

B. Feature Selection

Given the extensive variety of input features available, the adoption of Feature Selection proves to be a crucial step when involving data mining and machine learning, as this method can be essential for strategically choosing the ideal set of characteristics for analysis and prediction. For this challenge, we chose two advanced supervised techniques for Feature Selection: Filter-based Approach and Wrapper-based Approach.

[†]We would like to thank our Professors Glauco Gonçalves and Aldebaro Klautau for all their support and guidance throughout the GNN challenge 2023.

1) Filter-based Approach: Filter-based models are computationally faster compared to the other methods. The features are ranked based on their scores on various statistical tests between the specific feature and the output label. From this, we were able to verify the features that are most relevant to our model. To that end, we chose the Mutual Information Gain (MIG) algorithm, which measures the dependency between the variables based on an entropy estimation from k-nearest neighbors distances as described in [2].

2) Wrapper Approach: Because filters may not identify features that bring complementary information from others, we also tested Wrappers. More specifically we used the Exhaustive algorithm, from the Wrapper method, which searches for all possible combination of features, evaluates the model over each subset, and return the best score, which makes this method computationally expensive. The best subsets of features were selected based on `LinearRegression()` from the Sklearn library, and the metric*"negative_mean_squared_error"* was adopted. Moreover, all these algorithms are at *feature_selection.py* code.

The subset of features that received the best rankings by the Filter method was similar to the best subset selected by the Wrapper method, confirming that they are good features for our model. Table III presents the final subset of features used as input to our model, which were selected using Feature Selection techniques. In this sense, the features below are organized in descending order from the highest contribution to the lowest, as generated by the filter algorithm.

Features
flow_traffic, flow_ipg_var, flow_bitrate_per_burst, flow_p90PktSize, flow_packets, rate, flow_pkts_per_burts, flow_ipg_mean, flow_length, ibg

TABLE III: Subset of features used as input.

C. Z-score Normalization

The Z-score normalization was used in this work in place of the default feature normalization found in the model baseline, that is, *min-max* normalization. Usually, the Z-score approach can lead to a model more robust to outliers values than *min-max* considering the distribution of each feature, since the normalization parameters of a feature vector **x** is defined by the mean(μ) and the standard deviation(σ) of this array. Therefore given a sample x of an array, the normalization value is defined by $x_{norm} = \frac{x-\mu}{\sigma}$.

Therefore, to implement this normalization in our model, the method `get_mean_std_scores()` in the *std_train.py* was created inspired by the method `get_min_max_scores()`; to get the mean and the standard deviation of a given sample. After that, a few modifications were also necessary to adapt the *std_train.py*, to call this method and the *std_models.py* intrinsic classes to use this method and the returned parameters to perform the new normalization.

D. Attention Mechanism

Bronstein, et al. defined in [3] a narrow relation between attentional GNN models, such as Graph Attention Networks (GAT), with Message Passing Neural Networks (MPNN). In this sense, they express this relation in terms that MPNN represents a generalization of GAT models, where the main difference is how the *update* function is defined in each model. Thus, given this proposed generalization, it is utterly understandable to leverage these attention mechanisms by applying it inside of *update* function present in the MPNN model. Therefore, our idea relied exactly on the concept of exploring the attention layer simultaneously with the structure of multiple-stage message passing present in the baseline. Hence, considering that feature selection was already used, the attention mechanism would be another layer that provides a kind of fine-tuning to identify the importance of selected features, but in this case, not eliminating definitively, but decreasing or increasing their relevance from scores. In this sense, Equation

(1) dictates the process to compute these scores. This type of attention mechanism was proposed by [4] and it can be defined as an enhanced model previously thought of by [5]. The difference between them is the order in which each operation is applied.

$$\alpha_{uv} = \frac{\exp(\mathbf{a}^T \text{LeakyReLU}(\mathbf{W}[\mathbf{h}_u || \mathbf{h}_v]))}{\sum_{k \in \mathcal{N}_u} \exp(\mathbf{a}^T \text{LeakyReLU}(\mathbf{W}[\mathbf{h}_u || \mathbf{h}_k]))} \tag{1}$$

The reason for choosing Equation (1) instead of the attention equation defined by [5], is due to how the message phase is performed in the RouteNet Model. In this way, the concatenation operation($||$) is already used in the aggregation of interest node features h_u and its neighborhood node features h_k, such that $k \in \mathcal{N}_u$. Where $\mathcal{N}_u$ is the neighborhood of interest node u. With this aggregation provided by the baseline, is simpler just to multiply this result by a weight matrix $\mathbf{W}$ and perform, in the output of this operation, the non-linearity with the LeakyReLU. Furthermore, all these operations can be associated with a simple MultiLayer Perceptron, whose final output is called the attention coefficient for each feature. These coefficients are the score that defines the relevance of each feature; although each value has a different scale. Thus, for comparison purposes, it is necessary to normalize them; in this case, the Softmax function is used for that, and finally generates the attention score α as shown in Equation (1).

III. Tutorial Steps

A. Creating the Anaconda Virtual Environment

Considering that one has the *miniconda* or the *Anaconda* virtual environment already installed in a Linux machine, to recreate the environment used in our solution, the left command below should be used to create the environment properly and the right one to activate it.

`conda env create -f environment.yml` → `conda activate gnn_2023_m0b1us`

B. Training

Considering the current directory is the *m0b1us_model*, to perform the training, it is necessary to execute the command below. This command will train the CBR+MB model considering the dataset at `datasets/data_cbr_mb_13`. Where the `--ckpt-path` flag defines the weights directory name.

```
python3 std_train.py -ds CBR+MB --ckpt-path best_model
```

C. Prediction

Intended to create the predictions utilized in our best solution, the command below should be used. This command takes into account the CBR+MB model, the best weight `150-15.7196`, the training and testing datasets at `datasets/data_cbr_mb_13` and `dataset/data_test`, respectively.

```
python3 std_predict.py -ds CBR+MB --ckpt-path weights/150-15.7196 \
--tr-path datasets/data_cbr_mb_13_cv/0/training/ \
--te-path datasets/data_test/
```

References

[1] Miquel Ferriol-Galmés et al. "RouteNet-Fermi: Network Modeling With Graph Neural Networks". In: *IEEE/ACM Transactions on Networking* (2023), pp. 1–. DOI: 10.1109/tnet.2023.3269983.

[2] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical Review E* 69.6 (June 2004). DOI: 10.1103/physreve.69.066138.

[3] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. arXiv: 2104.13478 [cs.LG].

[4] Shaked Brody, Uri Alon, and Eran Yahav. *How Attentive are Graph Attention Networks?* 2022. arXiv: 2105.14491 [cs.LG].

[5] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML].