A Novel Network Delay Prediction Model with Mixed Multi-layer Perceptron Architecture for Edge Computing

Honglin Fang†, Peng Yu†*, Ying Wang†, Wenjing Li†, Fanqin Zhou†, and Run Ma§,
†State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
§State Grid Ningxia Electronic Power Co., LTD., China
*Corresponding author: yupeng_bupt@126.com

***Abstract*—Network delay is a crucial indicator for realizing delay-sensitive task offloading, network management, and optimization in B5G/6G edge computing networks. However, the delay prediction for edge networks becomes complicated due to diverse access strategies and heterogeneous services' storage, computing, and communication resource requirements. Current GNN-based delay prediction models such as RouteNet and PLNet lack the ability to express the complex associations between links and paths, so the predicted delay is not accurate. In this paper, we propose a novel end-to-end delay prediction model named MixerNet for edge computing, which is based on the mixed multi-layer perceptron (MLP). In this model, a mixed MLP architecture is applied to represent the association between links in the network topology and various paths. Observing that each link may have different effects on various paths, a weight matrix is then defined and multiplied by the path matrix to express it. Thus, a complete mapping frame from network characteristics (e.g., traffic intensity and routing schemes) to delay indicator is constructed. Finally, we perform extensive experiments on NSFNET and GEANT2 datasets and regard RouteNet as the baseline model. Experimental results show that MixerNet can accurately predict end-to-end delay results on various network topologies and the mean absolute error is merely about 0.36%. MixerNet also outperforms the baseline model in most evaluation indicators, especially the mean square error has a 3-fold decrease in NSFNET.**

***Index Terms*—Edge Computing, Delay Prediction, Multi-layer Perceptron**

I. Introduction

With the rapid development of communication technologies, beyond 5G (B5G) and 6G edge computing networks are expected to provide effective task offloading functions at the network edge to meet the rigorous delay requirements of heterogeneous services [1]. In edge networks, the delay indicator is regarded as a critical optimization indicator of service quality and network status [2], which is also used to evaluate the performance of edge computing task offloading [3] and routing schemes [4].

Therefore, the delay prediction model becomes an essential component in achieving dynamic network management, and optimization [5]. The role of the delay prediction model is to construct the exact mapping relationships between the network characteristics (e.g., traffic matrix, routing schemes, network topologies) and the per-path delay indicators. In addition, it can not only provide a real-time and zero-risk evaluation environment of network performance for various optimization strategies but also reduce the cost of the edge network performance monitoring [6]. Due to diverse access strategies and heterogeneous storage, computing, and communication resource requirements, the network topologies, and traffic features become more complex, increasing the difficulty of delay prediction.

In this context, several efforts have been devoted to constructing delay prediction models. Traditional models are mainly based on queuing theory [7] and make simple assumptions about network traffic distribution and probabilistic routing. These assumptions may be different from the actual network properties. They may lead the model to fail to fine-grained express the mapping relationship from network characteristics to delay indicators [6].

Recently, deep learning (DL) models demonstrate superior representation ability and efficient computational speed, which is a well-suited method for building highly accurate delay prediction models. Deep-Q [8] is proposed to express the QoS metrics from traffic, RouteNet [6] leverages graph neural network (GNN) to make accurate per-path delay estimations, PLNet [9] predicts network performance distribution with path-link GNN, xNet [10] additionally introduces queues and flow features to obtain the delay predictions.

However, these DL-based models with diverse types of GNN may lack the ability to express the inner correlations that may exist between any two links. PLNet [9] takes the sum of the corresponding link features as updated path features, so it does not take into account the effect between links. RouteNet [6] and xNet [10] apply gated recurrent unit (GRU) [11] to orderly aggregate link features, so it can only simply express the correlation between pre-order links and post-order links.

In this paper, we propose MixerNet, a novel delay prediction model with mixed multi-layer perceptron (MLP) architecture [12] to fine-grained express complex correlations between network status and the delay indicator. Firstly, observing that each link may have different impacts on various paths, a weight layer is designed to sum the corresponding link features from all path features containing it. Then the link update block and path update block are defined by the mixed MLP

978-3-903176-51-5 © 2022 IFIP

structure, which contains one *relation-mixing* MLP and one *feature-mixing* MLP to extend the receptive field, and widely express potential relationships within paths and links. Finally, we construct a complete mapping frame from network characteristics (e.g., traffic intensity and routing schemes) to the delay indicator and perform extensive experiments to evaluate MixerNet on NSFNET and GEANT2 network delay prediction datasets. Experiment results demonstrate that the structure of MixerNet can be better and fully utilized in various network topologies and traffic intensities. MixerNet achieves 3.43 times higher prediction accuracy than RouteNet in NSFNET and enables it to better adapt to the congested traffic environment.

II. Related Work

Delay prediction models aim to construct the internal correlations of several network characteristics to represent the per-path network delay. Previous research efforts are being devoted to predicting the network delay.

Traditional models attempt to predict delay results by queuing theory [7] or network calculus [13] methods. However, these models often make simple assumptions like the traffic satisfies poisson distribution, and packets received by different nodes are independent, which may be non-realistic network characteristics and will make the traditional models fail to fulfill their accurate prediction expectations.

As mentioned in section I, the network delay may be impacted by not only traffic distributions but also many network states such as network topologies, routing schemes, and link bandwidth. To accurately predict delay, the model needs to adequately represent the complex relationships inherent in these network state metrics [10]. However, the numerous network state indicators and the intrinsic correlations between these indicators are too complex to model directly. Overall, the prediction results of the traditional model are limited by the accuracy and lack the ability to generalize the routing configurations, and the network topologies [6].

Following the powerful expression of deep learning, researchers recently attempt to predict delay by applying various neural network architectures like recurrent neural networks, graph neural networks, and fully-connected neural networks. Deep-Q [8] designs a deep generative network with LSTM to infer QoS metrics with real-time traffic conditions as input. RouteNet [6] models path traffic and link bandwidth respectively and utilizes message passing neural network (MPNN) to express the inherent relationships between links and paths. RouteNet predicts network performance metrics with traffic distribution, routing schemes, and network topologies.

Then, PLNet [9], a variant of RouteNet, updates the link and path features by bipartite path-link graph and message networks to improve the delay inference speed. xNet [10] represents various network characteristics and extends MPNN architectures to infer data center network performance metrics.

However, these DL-based models lack the ability to express the complex correlations between paths and links. Deep-Q only considers the per-path traffic effects and is unable to represent link bandwidth features that may affect the prediction results. RouteNet and its variants do not take into account the feature passing process between links when updating link features. Moreover, they assume all paths have the same impact on links, meaning that they are hard to express the different interaction between links and paths.

III. Edge Computing Scenario

A simple application scenario of delay prediction model in edge computing networks is illustrated in Fig. 1, which contains one decision-making service, six base stations with edge computing servers, seven user equipments with various delay-sensitive computing tasks like video decoding, AI inference, and VR rendering. The decision-making service, including the optimizer module and the delay prediction model, ensures the stable operation of heterogeneous network services [14]. The end-to-end delay results provided by the delay prediction model can assist the optimizer to offload and manage computing tasks.

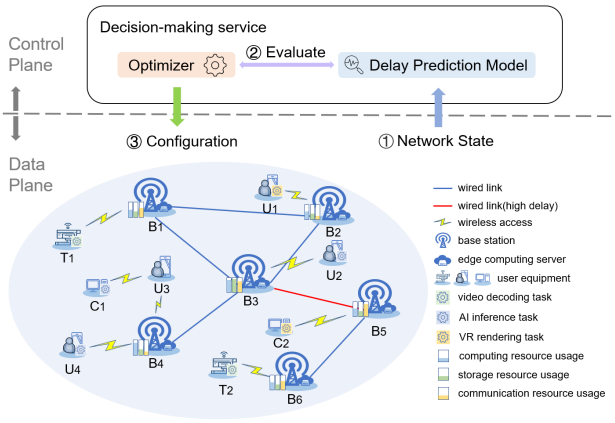


Fig. 1. Illustration of delay prediction model in edge computing scenario

For example, the VR rendering task on C_2 is deployed on B_3 edge computing servers after being transmitted through B_5 node. When the delay indictor between B_3 and B_5 is abnormal, the decision-making service can timely update the deployment strategies and offload the tasks of C_2 to B_5 or B_6 node to ensure the delay requirements of VR rendering tasks.

In summary, the delay prediction model can be applied to three optimization problems to improve the edge computing network performance.

(1) low-cost network performance monitoring and simulation. Traditional edge networks need physical monitoring probes and real-time interfaces to obtain network performance information. The delay prediction model can accurately infer the end-to-end delay from network status in real-time without the help of the high-cost delay perception [10]. Besides, the model can also simulate and test the network performance in various non-realistic environments to assist in optimizing the network structure.

(2) delay-sensitive tasks offloading. When user equipments initiate edge computing tasks like camera T_1 needs to upload photos or videos for face recognition, the SDN controller

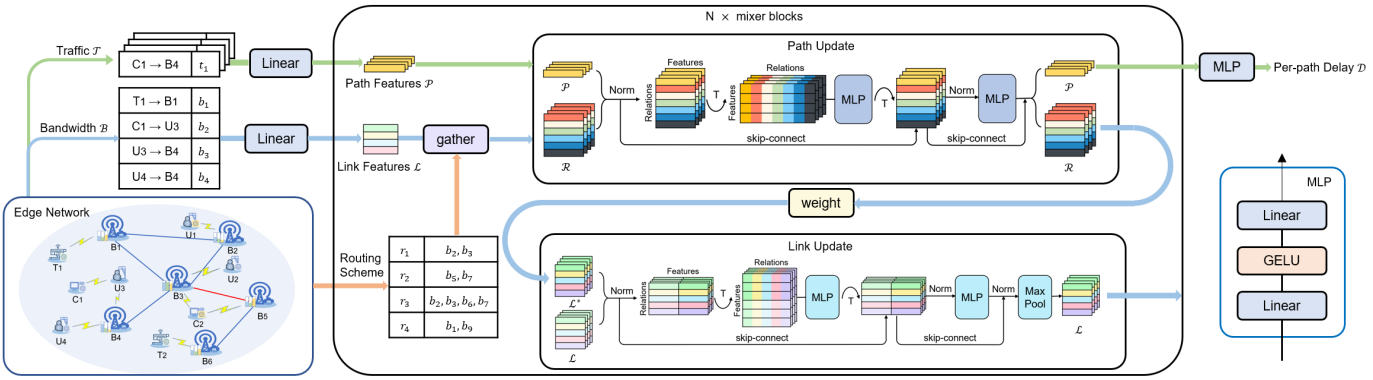


Fig. 2. The MixerNet model architecture. $\mathcal{P}$ is the path feature matrix, and $\mathcal{L}$ is the link feature matrix. $\mathcal{B}$ is the bandwidth matrix of all links, $\mathcal{T}$ is the traffic matrix of all paths, and $\mathcal{D}$ is the per-path delay. The MixerNet is composed of a stack of N mixer blocks. Each link or path update block contains one *relation-mixing* MLP and one *feature-mixing* MLP. Each MLP includes a GELU activation function and two linear layers, which is also named fully-connected layer. Other components consist: skip connections, and layer norms.

needs to select an edge computing node from E_1 and E_2 to offload the computing task. The task offloading algorithm will comprehensively consider the end-to-end delay from T_1 to E_1 and E_2 with the delay prediction model and other network features to select a specific unloading node.

(3) zero-risk digital twin network environment. Recently, numerous network optimization models start to leverage deep reinforcement learning (DRL). However, the network configuration generated by DRL may be unstable and perform unreliably. To avoid unpredictable network risks and service failures, the delay prediction model can aid in creating a zero-risk digital twin network environment for DRL training and evaluation, then improve the reliability of network optimizers.

IV. MixerNet: Mixed MLP-based Delay Prediction Model

In this section, we first express the representation notations of network characteristics and delay. Then, a novel delay prediction model architecture is demonstrated and identified. Finally, we illustrate in detail the mixed MLP structures that are applied to update links and paths features.

A. Notation

An edge computing network N can be represented as $N = \{\mathcal{L}, \mathcal{P}, \mathcal{R}\}$. $\mathcal{L} = \{l_1, l_2, \ldots, l_i\}$ is the set of links and i is the number of links. $\mathcal{P} = \{p_1, p_2, \ldots, p_k\}$ is the set of end-to-end paths and k is the number of paths. $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$ is the set of path-specific link matrices according to routing schemes. Each link matrix contains an end-to-end sequence of link features for the corresponding path. In this paper, the network characteristics and delay can be represented as $S = \{\mathcal{B}, \mathcal{T}, \mathcal{D}\}$, where $\mathcal{B} = \{b_1, b_2, \ldots, b_i\}$ is the set of link bandwidth, $\mathcal{T} = \{t_1, t_2, \ldots, t_k\}$ is the set of path traffic, $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ is the set of path delay.

B. Model Architecture

As mentioned above, all these network characteristics and performance can be expressed by links or paths. For instance, the bandwidth features are represented by link attributes, delay, and traffic features are regarded as path attributes. Intuitively, the complex internal correlations between various network characteristics can be modeled by the relationships between network links and paths, which is also applied in RouteNet [6] and its variants [9], [10].

To fully express inner correlations between links and paths, we extend the modeling framework of RouteNet and design a novel delay prediction model named MixerNet. As shown in Fig 2, MixerNet is composed of a stack of N identical mixer blocks and takes the routing schemes, links, and paths features as the input.

MixerNet first applies the linear layer (fully-connected layer) to embed the per-path traffic and per-link bandwidth in different vector spaces. Each bandwidth or traffic value is projected to a desired *hidden dimension* c. The number of links in the network is defined as i, and the path number is represented by k.

In the initialization process, we select the traffic value as the initial path feature and regard the bandwidth value as the initial link feature. To express the relationship between path and corresponding links, a gather layer is designed and orderly splices the links contained in each path into path-specific link matrices $\mathcal{R}$ according to routing schemes. In Fig. 2, r_1 is the corresponding link matrix of path $p_1 : C_1 \rightarrow B_4$, which can be represented as $r_1 = [b_2, b_3]$ when the routing path is $C_1 \rightarrow U_3 \rightarrow B_4$.

Then a well-designed path update block (§IV-C) is applied to learn the relationships between path features $\mathcal{P}$ and corresponding link matrices $\mathcal{R}$. After that, we weight all updated path-specific link matrices to express the differential effect of links under different paths. Then all weighted link matrices $\mathcal{R}^*$ are added according to the link index to form new link features. We also define a novel link update block (§IV-C) which is similar to our path update block to represent correlations between any two links. The entire process of updating paths and links is named as a mixer block. The performance of the delay prediction model can be improved by stacking mixer block multiple times. Finally, the readout layer, consisting

of the MLP architecture, takes the last-update path features to produce the per-path delay results. Algorithm 1 further describes the internal framework and feature update process of MixerNet.

Algorithm 1 Mixed MLP-based delay prediction model, MixerNet.

Input: network characteristics $N_s = \{\mathcal{B}, \mathcal{T}\}$
Output: per-path delay $\mathcal{D}$

// Initialize link and path states.
$\mathcal{L}^0 = f_1(\mathcal{B})$, $\mathcal{P}^0 = f_2(\mathcal{T})$
for each n in range(N) **do**
  for each path k in $\mathcal{P}$ **do**
    Gather route features $r_k^{n-1} = \gamma(\mathcal{L}^{n-1})$
    Update $p_k^n, r_k^n = \text{Path-MLP}(p_k^{n-1}, r_k^{n-1})$
  end for
  $w^{n-1} = f_3(\mathcal{P}^n)$
  $\mathcal{L}^{n*} = \phi(w^{n-1} \cdot \mathcal{R}_p^n)$
  $\mathcal{L}^n = \text{Link-MLP}(\mathcal{L}^{n-1}, \mathcal{L}^{n*})$
end for
$\mathcal{D} = \text{Readout}(\mathcal{T}^N)$

As shown in Algorithm 1, f_1 and f_2 are fully-connected layers to initialize the link and path features, respectively. f_3 is a fully-connected layer to represent per-path weight from path features. γ is the gather function to build link features into path-specific link matrices according to special routing schemes. ϕ is the aggregate function for each link to sum the corresponding link features from all the routing matrices.

C. Mixed MLP for Path and Links

RouteNet and its variants simply employ GRU to update path and link features, which makes them only consider correlations between adjacent context links and lack of global receptive field. Moreover, it makes these models require strict input format and hard to execute in parallel.

To break these restrictions, we refer to the MLP-Mixer architecture [12] and design novel link and path update blocks with mixed MLPs, respectively. The mixed MLP structure mainly contains two types of layers: one MLP applied to relation channels to "mixing" the across features, and one MLP applied to feature channels to "mixing" the inner features. The motivation behind this structure is to expand the information exchange capability between various network characteristics. Fig. 2 summarizes these mixed MLP structures.

To conveniently describe the dimensions of various variables in the following, the maximum link number in link matrices $\mathcal{R}$ is defined as j. In the path update block, for path k, the path features $p_k \in \mathbb{R}^{c\times 1}$ and path-specific link matrix $r_k \in \mathbb{R}^{c\times j}$ are concatenated into a tensor matrix named path matrix $x_k \in \mathbb{R}^{c\times(j+1)}$. As shown in Fig. 2, the path update block include two types of MLPs. The first one is *relation-mixing* MLP: it performs on the relation channels and is shared across all features. The second one is *feature-mixing* MLP: it performs on feature channels and is shared across all relations. Each MLP consists of two fully-connected layers and GELU activation layers. The specific calculation process of n-th path update block can be defined as follows:

$$\begin{aligned} x_k^{n-1} &= \text{concat}(p_k^{n-1}, r_k^{n-1}), \\ x_k^* &= x_k^{n-1} + (W_2^p \cdot \theta(W_1^p \cdot \sigma(x_k^{n-1})^\top))^\top, \\ x_k^n &= x_k^* + W_4^p \cdot \theta(W_3^p \cdot \sigma(x_k^*)), \\ p_k^n, r_k^n &= \kappa(x_k^n), \end{aligned} \tag{1}$$

where concat$(\cdot)$ is the concatenation function, κ splits x_k^n from the first row into new path features p_k^n and new link matrix r_k^n. σ is the LayerNorm function [15], and θ is the element-wise GELU activation [16]. $W_1^p, W_2^p, W_3^p, W_4^p$ are the tensor matrices to make linear transformations to the path matrix. The $W_1^p \in \mathbb{R}^{\mu(j+1)\times(j+1)}$ and $W_2^p \in \mathbb{R}^{(j+1)\times\mu(j+1)}$ are combined into *relation-mixing* MLP, $W_3^p \in \mathbb{R}^{\mu c\times c}$ and $W_4^p \in \mathbb{R}^{c\times\mu c}$ are included in the *feature-mixing* MLP, where μ is the rate of hidden dimension. In MixerNet, the hidden rate μ is set to 0.5 to reduce the overfitting risk and model parameters.

In n-th link update block, the link features from the previous link update block $\mathcal{L}^{n-1} \in \mathbb{R}^{c\times i}$ and the link features from the weight layer $\mathcal{L}^{n*} \in \mathbb{R}^{c\times i}$ are concatenated into the input matrix $u^{n-1} \in \mathbb{R}^{2c\times i}$. The n-th link update block can be defined as follows:

$$\begin{aligned} u^{n-1} &= \text{concat}(\mathcal{L}^{n-1}, \mathcal{L}^{n*}), \\ u^* &= u^{n-1} + (W_2^l \cdot \theta(W_1^l \cdot \sigma(u^{n-1})^\top))^\top, \\ u^n &= u^* + W_4^l \cdot \theta(W_3^l \cdot \sigma(u^*)), \\ \mathcal{L}^n &= \varrho(u^n), \end{aligned} \tag{2}$$

where concat$(\cdot)$ is the concatenation function, ϱ is maxpooling function with a scale of two, and $W_1^l, W_2^l, W_3^l, W_4^l$ are the tensor matrix to make linear transformations to the link matrix. The $W_1^l \in \mathbb{R}^{\eta i\times i}$ and $W_2^p \in \mathbb{R}^{i\times\eta i}$ are defined as *relation-mixing* MLP, $W_3^p \in \mathbb{R}^{\eta c\times c}$ and $W_4^p \in \mathbb{R}^{c\times\eta c}$ belong to *feature-mixing* MLP, where η is the rate of hidden dimension. In MixerNet, the hidden rate η is set to 0.5 to reduce parameters and avoid the phenomenon of sparse distribution of features.

V. Simulation Results and Discussions

In this section, we apply two mainstream delay prediction datasets to evaluate the prediction accuracy and generality of MixerNet. First, We reproduced RouteNet and regard it as the baseline model. Then extensive experiments are used to compare the delay prediction results and the distribution of prediction errors of RouteNet and MixerNet with various traffic intensities, network topologies, and routing schemes. Finally, all simulation results show that MixerNet can produce more accurate and stable delay prediction results. All simulations are implemented with the Pytorch framework [17] and run on Tesla V100 GPUs.

A. Simulation setup

We perform experiments on 14-node NSFNET [18] and 24-node GEANT2 [19] datasets from the popular available Knowledge-Defined Networking (KDN) project, which are

both generated by packet-level simulator OMNeT++ [20]. Details of these datasets are shown in Table I.

TABLE I
STATISTICS OF NETWORK DATASETS USED IN SIMULATIONS

Network	Nodes	Links	Max path length	Avg. path length
NSFNET	14	42	4	2.14
GEANT2	24	74	7	2.92

These KDN datasets assume that the network traffic can be randomly generated by every end-to-end node pair [6]. Although in the entire mobile communication network, most traffic data is concentrated between the radio access network and the core network, the edge computing scenario will make the traffic distribution more dispersed [9]. Therefore, we believe that the network characteristics distributions of NSFNET and GEANT2 datasets are similar to actual edge computing scenarios. The simulation results are also reliable and convincing in the edge computing scenario.

B. Evaluation indicators and loss function

To describe the calculation process of the evaluation indicators, we define the real per-path delay of the network in a certain time slice as $D = [d_1, d_2, \ldots, d_k]$, and define the model prediction delay as $D' = [d'_1, d'_2, \ldots, d'_k]$. To measure the prediction error and accuracy, we adopt the following four evaluation indicators as well as recent works:

- Mean squared error (MSE):

$$\mathrm{MSE} = \frac{1}{k}\sum_{i=1}^{k}(d_i - d'_i)^2, \tag{3}$$

  where k is the path number of the network.
- Mean absolute error (MAE):

$$\mathrm{MAE} = \frac{1}{k}\sum_{i=1}^{k}|d_i - d'_i|, \tag{4}$$

  where $|\cdot|$ means the absolute value.
- Pearson correlation coefficient (PCC):

$$\mathrm{PCC} = \frac{\mathbb{E}[(D - \mathbb{E}[D])(D' - \mathbb{E}[D'])]}{\sigma_D \sigma_{D'}}, \tag{5}$$

  where $\mathbb{E}$ is the expectation, σ_D and $\sigma_{D'}$ are the standard deviation of D and D', which is defined as follows:

$$\sigma_D = \sqrt{\mathbb{E}[(D - \mathbb{E}[D])^2]} = \sqrt{\mathbb{E}[D^2] - (\mathbb{E}[D])^2}. \tag{6}$$

- Mean absolute percentage error (MAPE):

$$\mathrm{MAPE} = \frac{1}{k}\sum_{i=1}^{k}\left|\frac{d_i - d'_i}{d_i}\right|. \tag{7}$$

MSE is the most popular evaluation indictors to measure the mean error between real delay and prediction result. However, when the error is much less than 1, the square operation of MSE will make the errortoo small to be measurable, so we also apply MAE indictor to reduce this effect. In addition, we employ PCC to measure the linear correlation between predictions and ground truth. We also use MAPE to normalize the prediction error of different delay scales.

A highly accurate delay prediction model will show high PCC and low MSE, MAE, and MAPE. In MixerNet, we use the MSE as the loss function like RouteNet and its variants because it is more robust and can achieve better performance than other evaluation indicators [9].

C. Implementation

Since PLNet [9] and RouteNet [6] achieve a similar accuracy level and xNet [10] is not suitable for the edge computing scenario, we select the RouteNet as the baseline model.

We use the Adam optimizer to minimize the MSE with the learning rate of 0.001. In addition, we further improve the prediction performance by reducing the learning rate by half when the validation MSE indicator stops decreasing. Validation is performed every 5000 training steps. To sufficiently train MixerNet, the early stopping mechanism is used to monitor the MAPE metric and stop the training when no improvement is observed for 10 validation steps.

To balance the performance and inference speed, and ensure the fairness in comparision with RouteNet, we set the stacking times of the mixer block to 8. We also apply a grid search algorithm in the NSFNET dataset with the MAPE metric as the optimization objective to tune hyper-parameters. The well-tuned hyper-parameters are summarized as follows: batch size $b = 32$, dimension of path and link features $d = 128$, dimension of readout MLP $\delta = 256$. Finally, we conduct extensive experiments on both NSFNET and GEANT2 datasets using these well-tuned hyper-parameters.

D. Comparison of evaluation indicators

Before the training process, these datasets are split into the train and validation sets in 8:2, respectively. Each set contains numerous network simulation results with 9, 12, and 15 traffic intensities. We report the delay prediction performance of RouteNet and MixerNet on two popular network datasets under various traffic intensities. The experiment results of MixerNet and RouteNet are shown in Table II.

TABLE II
DELAY PREDICTION RESULTS ON TWO NETWORKS. THE BEST RESULTS ARE IN BOLD

RouteNet/MixerNet Dataset	Traffic	MSE 10^{-4}	MAE 10^{-3}	PCC %	MAPE 10^{-2}
NSFNET	9	0.20/**0.12**	3.24/**2.56**	99.93/**99.96**	1.31/**1.05**
	12	0.54/**0.19**	4.55/**3.10**	99.94/**99.98**	1.31/**0.97**
	15	3.06/**0.56**	9.87/**4.93**	99.96/**99.99**	1.54/**0.97**
	all	1.33/**0.30**	6.04/**3.59**	99.96/**99.99**	1.39/**0.99**
GEANT2	9	0.26/**0.17**	3.30/**2.76**	99.93/**99.95**	1.73/**1.48**
	12	1.03/**0.38**	5.04/**3.42**	99.95/**99.98**	1.69/**1.31**
	15	3.06/**0.84**	8.74/**4.87**	99.97/**99.99**	1.83/**1.22**
	all	1.48/**0.47**	5.74/**3.70**	99.97/**99.99**	1.75/**1.34**

As shown in Table II, MixerNet outperforms the baseline model in all evaluation indicators under all traffic intensities of the two delay prediction datasets. Compared with RouteNet, the MSE metric is 3.43 times lower in NSFNET and 2.15 times lower in GEANT2, the MAE metric is reduced by 62%,

and the MAPE is reduced by 36%. In addition, the PCC indicator of MixerNet reached 99.99% on both datasets. The experiment results show that MixerNet has a more accurate delay prediction ability than RouteNet.

We also notice that the prediction error becomes progressively more prominent as the traffic intensity increases from 9 to 15, and the error of RouteNet rises much more than MixerNet. On NSFNET, the MSE metric of RouteNet increases by 14.3 times, and it only increases by 3.7 times in MixerNet; on GEANT2, RouteNet increases by 10.8 times, and MixerNet increases by only 3.9 times. It is intuitive that MixerNet is more stable and reliable than RouteNet and still has higher prediction accuracy in the high traffic load scenario.

E. Comparison of prediction error distributions

In the network scenario, the delay prediction model with a more concentrated distribution of prediction error represents better stability. A small number of results with large prediction errors may bring significant security risks to the network management and services operation [6]. However, these wrong results can not be reflected in the evaluation indicators when numerous well predictions exist.

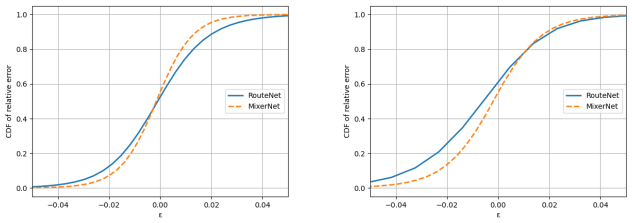
Hence, we report the cumulative distribution function (CDF) of the relative error on two datasets over all the evaluation samples to compare the model performance comprehensively. The relative error is defined as follows:

$$\varepsilon = \frac{d - d'}{d}, \tag{8}$$

where d is the real delay result and d' represent the prediction result.

The CDF result is illustrated in Fig. 3. The horizontal axis represents the relative error value, and the vertical axis represents the proportion of relative error results that are less than specified ε. The cumulative distribution function of a relative error variable E is defined as:

$$F_E(\varepsilon) = P(E < \varepsilon). \tag{9}$$



(a) CDF of NSFNET (b) CDF of GEANT2

Fig. 3. The CDF result of the relative error ε.

The CDF value of an accurate model will approach 0 when $\varepsilon < 0$, and approach 1 when $\varepsilon > 0$. Fig. 3 shows that MixerNet can provide more accurate and centralized delay prediction results, which is closer to the ground truth than the predictions of RouteNet. Moreover, we also notice that the relative error distributions of RouteNet are unbalanced, especially on the GEANT2 dataset. It demonstrates that more than 60% of the prediction results produced by RouteNet are higher than the real delay value on GEANT2. However, this abnormal phenomenon is not evident in MixerNet, meaning that the fine-grained modeling of network characteristics by mixed MLP architecture in MixerNet can effectively alleviate the imbalance in relative errors.

F. Comparison on various traffic intensities

From the above discussion, we notice that the prediction accuracy of both MixerNet and RouteNet go down when per-path traffic intensity rises. To further compare the delay prediction performance of these models, we extensively report the prediction error distributions of several traffic intensities.

The percentage error ν is regarded as the indictor when the traffic intensity is set to 9, 12, 15. The percentage error is the absolute value of the relative error and is defined as:

$$\nu = |\varepsilon| = |\frac{d - d'}{d}|, \tag{10}$$

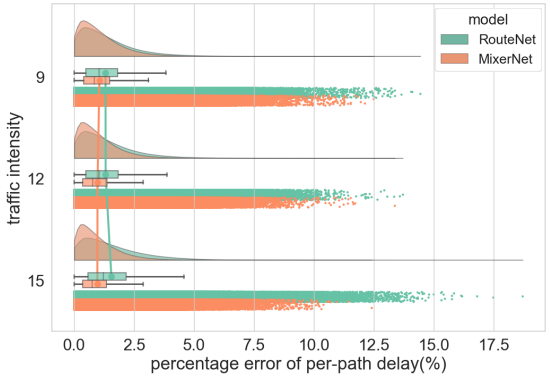
The raincloud plot [21] is applied to visualize and compare the distribution of prediction errors for different traffic intensities, which attempts to provide an intuitive, statistically robust picture of the data distribution.

In essence, the raincloud plot contains four kinds of subgraphs. The first one is a cloud plot: it is expressed by the half of a violin plot to estimate the kernel density for each prediction error value. The second one is a umbrella plot: it is generated by a box plot without outliers to express the overall distribution of prediction errors. The third is a rain plot: it is produced by a raw scatter plot to visualize the location of the furthest outliers in massive data. The last is a thunder plot: it is a traditional line plot connecting the mean values across each category. Using the raincloud plot, we present the density estimates of the percentage errors for various traffic intensities in NSFNET and GEANT2.

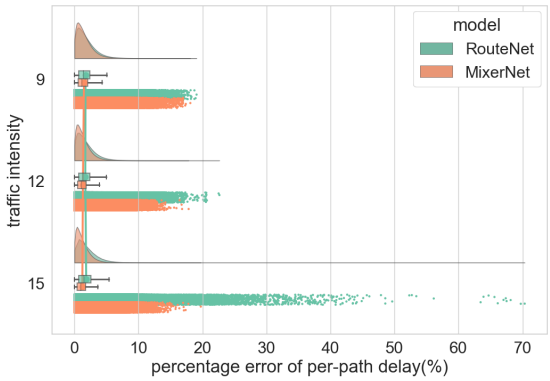
As shown in Fig. 4, the distributions of percentage errors for MixerNet at all three traffic intensities are closer to 0 than RouteNet, thus illustrating that MixerNet enables to infer more accurate and stable prediction results. In Fig. 4(a), we notice that MixerNet maintains a lower and more concentrated percentage error distribution in all traffic intensities, while it is dispersed with the traffic density increases in RouteNet. On the rain plot in Fig. 4(b), the maximum value of outlier points in MixerNet is less than 20%. In constrast, the maximum percentage error of RouteNet is larger than 70%, thus indicating that the model structure of MixerNet enables it to provide accurate and stable delay prediction results under different traffic intensities environments.

VI. Conclusion

In this paper, we first provide various application scenarios of the delay prediction model in edge computing networks. Then, we propose a novel end-to-end delay prediction model named MixerNet, which is based on the mixed MLP architecture. In the well-designed link update block and path update block, a novel mixed MLP architecture is applied



(a) The raincloud plot of NSFNET



(b) The raincloud plot of GEANT2

Fig. 4. The raincloud plot of MAPE under different traffic intensities.

to extend the receptive field of path and link during the feature update process and enhance the representation ability of MixerNet by modeling the potential correlations between links and paths. In addition, the weight layer is designed to express the diverse roles of each link in various paths. Finally, we conduct comparative experiments for MixerNet and RouteNet on two mainstream network delay prediction datasets. Experiment results illustrate that the MSE metric of MixerNet is 3.43 times lower in NSFNET dataset and 2.15 times lower in GEANT2 dataset.

In future work, we plan to test the prediction performance of MixerNet for other network performance indicators (e.g., per-path jitter and packet loss). Furthermore, we will explore the general ability of the mixed MLP architecture in the network delay prediction task.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No.61901054).

References

[1] Y. Bai, L. Chen, L. Song, and J. Xu, "Risk-aware edge computation offloading using bayesian stackelberg game," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1000–1012, 2020.

[2] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, 2021.

[3] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 2992–3005, 2021.

[4] H. Xu, Z. Yu, X.-Y. Li, L. Huang, C. Qian, and T. Jung, "Joint route selection and update scheduling for low-latency update in sdns," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3073–3087, 2017.

[5] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Ma'ruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, p. 2–10, 2017.

[6] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[7] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, P. Andres-Maldonado, and J. M. Lopez-Soler, "Performance modeling of softwarized network services based on queuing theory with experimental validation," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1558–1573, 2021.

[8] S. Xiao, D. He, and Z. Gong, "Deep-q: Traffic-driven qos inference using deep generative network," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, ser. NetAI'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 67–73.

[9] Y. Kong, D. Petrov, V. Räisänen, and A. Ilin, "Path-link graph neural network for ip network performance prediction," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 170–177.

[10] M. Wang, L. Hui, Y. Cui, R. Liang, and Z. Liu, "xnet: Improving expressiveness and granularity for network modeling with graph neural networks," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 2028–2037.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[12] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 24 261–24 272.

[13] F. Ciucu and J. Schmitt, "Perspectives on network calculus: No free lunch, but still good value," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 311–322.

[14] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2833–2849, 2020.

[15] J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv*, vol. abs/1607.06450, 2016.

[16] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *ArXiv*, vol. abs/1606.08415, 2016.

[17] A. Paszke and et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[18] X. Hei, J. Zhang, B. Bensaou, and C.-C. Cheung, "Wavelength converter placement in least-load-routing-based optical networks usinggenetic algorithms," *J. Opt. Netw.*, vol. 3, no. 5, pp. 363–378, May 2004.

[19] F. Barreto, E. Wille, and L. Nacamura, "Fast emergency paths schema to overcome transient link failures in ospf routing," *International Journal of Computer Networks & Communications*, vol. 4, 04 2012.

[20] A. Varga, "The omnet++ discrete event simulation system," *Proc. ESM'2001*, vol. 9, 01 2001.

[21] M. Allen, D. Poggiali, K. Whitaker, T. R. Marshall, and R. A. Kievit, "Raincloud plots: a multi-platform tool for robust data visualization," *Wellcome open research*, vol. 4, 2019.