TransMUSE: Transferable Traffic Prediction in MUlti-Service Edge Networks

Luyang Xu[a,b,c,*], Haoyu Liu[b], Junping Song[a], Rui Li[d], Yahui Hu[e], Xu Zhou[a,*] and Paul Patras[b]

[a]Computer Network Information Center, Chinese Academy of Sciences, Building No.2, 4, Zhongguancun Nansijie, Haidian District, Beijing, 100190, Beijing, China
[b]School of Informatics, The University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
[c]University of Chinese Academy of Sciences, No.19(A) Yuquan Road, Shijingshan District, Beijing, 100049, China
[d]Samsung AI Center, 50 Station Road, Cambridge, CB1 2JH, United Kingdom
[e]China university of mining and technology-Beijing, Ding No.11 Xueyuan Road, Beijing, 100083, China

ARTICLE INFO

Keywords:
Edge Model Transfer
Multi-service Traffic Prediction
Service Clustering

ABSTRACT

The Covid-19 pandemic has forced the workforce to switch to working from home, which has put significant burdens on the management of broadband networks and called for intelligent service-by-service resource optimization at the network edge. In this context, network traffic prediction is crucial for operators to provide reliable connectivity across large geographic regions. Although recent advances in neural network design have demonstrated potential to effectively tackle forecasting, in this work we reveal based on real-world measurements that network traffic across different regions differs widely. As a result, models trained on historical traffic data observed in one region can hardly serve in making accurate predictions in other areas. Training bespoke models for different regions is tempting, but that approach bears significant measurement overhead, is computationally expensive, and does not scale. Therefore, in this paper we propose TransMUSE (Transferable Traffic Prediction in MUlti-Service Edge Networks), a novel deep learning framework that clusters similar services, groups edge-nodes into cohorts by traffic feature similarity, and employs a Transformer-based Multi-service Traffic Prediction Network (TMTPN), which can be directly transferred within a cohort without any customization. We demonstrate that TransMUSE exhibits imperceptible performance degradation in terms of mean absolute error (MAE) when forecasting traffic, compared with settings where a model is trained for each individual edge node. Moreover, our proposed TMTPN architecture outperforms the state-of-the-art, achieving up to 43.21% lower MAE in the multi-service traffic prediction task. To the best of our knowledge, this is the first work that jointly employs model transfer and multi-service traffic prediction to reduce measurement overhead, while providing fine-grained accurate demand forecasts for edge services provisioning.

1. Introduction

Edge computing pushes computation and data storage closer to the user, thereby improving response times and saving communication bandwidth, while serving multiple applications simultaneously, e.g., video streaming, gaming, content delivery, etc. As people work increasingly more often remotely following the Covid-19 outbreak and require network support for different services, the edge computing paradigm is witnessing growing uptake.

In order to optimise user experience and operational costs, infrastructure providers have been pursuing dynamic provisioning of network resources based on predictions of user demand [1]. Previous efforts in tackling network traffic prediction frequently exploit the ability of deep neural networks (DNNs) to learn complex patterns from historical data [2, 3, 4, 5, 6, 7]. However, existing solutions either require training one dedicated model for each geographic region and hence have limited transferability (which is of paramount importance in reducing computational costs and the environmental footprint of training DNNs) [2, 3, 4, 5], or disregard essential correlations among services [6, 7].

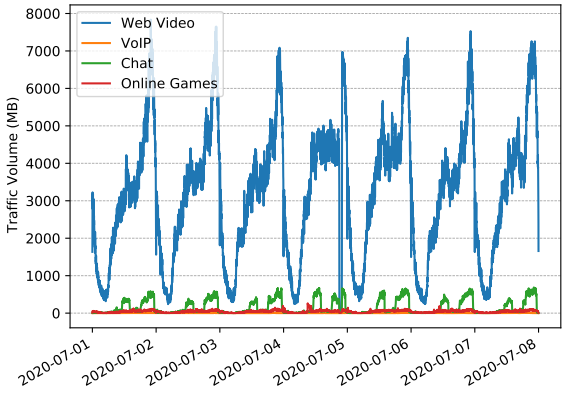


Figure 1: A snapshot of the volume of traffic consumed by four different services as observed at an edge node in a network deployment in Sichuan (China) over one week.

*Corresponding author
xuluyang@cnic.cn (L. Xu); haoyu.liu@ed.ac.uk (H. Liu); songjunping@cnic.cn (J. Song); rui.li@samsung.com (R. Li); huyahui@cumtb.edu.cn (Y. Hu); zhouxu@cnic.cn (X. Zhou); Paul.Patras@ed.ac.uk (P. Patras)
ORCID(s):

In practical large-scale network deployments *(i)* per-service patterns are often distinct within a region, as exemplified in Figure 1, while *(ii)* certain areas may exhibit similar characteristics that would allow for direct transfer of models among them, without the need of retraining. These key observations are confirmed by our analysis of a real-world network traffic dataset collected in a major city in Sichuan province, China, serving 2.6 million users, spanning 6.3 square kilometers, and comprising eight edge nodes. This motivates us to propose TransMUSE, a transferable traffic prediction framework in multi-service edge networks, which first groups edge-nodes according to per-service statistical features. Within each cohort, reference neural models are chosen and trained on data collected only in the region with the highest overall traffic consumption, which can be then transferred to other group members. As reference model, we put forward a Transformer-based [8] Multi-service Traffic Prediction Network (TMTPN). Furthermore, we propose WK-means, a service clustering algorithm based on Wasserstein distance to categorize services according to their similarity. We train separately a TMTPN model for each service cluster to boost prediction performance at a regional level. Finally, the reference models are transferred to other regions directly, without adaptation.

Our proposed model transfer framework, TransMUSE, provides a comprehensive and cost-effective solution for traffic prediction in multi-service edge networks. The key advantages of TransMUSE are as follows: *(i)* it provides a model transfer approach among edge nodes to reduce measurement and computational overhead without compromising prediction accuracy – compared with training a model individually on local data for each edge node, TransMUSE exhibits imperceptible performance degradation, with only 1.7% and 0.26% higher MAE and RMSE, respectively; *(ii)* the proposed TMTPN takes service correlation into consideration to further reduce overhead and the energy that would have otherwise been required to maintain a separate prediction model for each service; our experiments demonstrate that TMTPN outperforms the state-of-the-art MTNet benchmark [2] on the multi-service traffic prediction task by 18.74% and 18.49%, in terms of MAE and respectively RMSE; *(iii)* the WK-means service clustering tackles both model under-fitting and speed of convergence, improving the TMTPN prediction performance, as it attains 17.59% and 27.89% lower MAE and respectively RMSE, as compared to predicting without prior service clustering. To the best of our knowledge, TransMUSE is the first multi-service traffic forecasting solution for edge networks that leverages model transfer and service clustering to achieve high accuracy at a low measurement cost.

The rest of the paper is organised as follows. The multi-service prediction problem is formalised in Section 2. The proposed TransMUSE framework is discussed in detail in Section 3. Section 4 provides exhaustive experimental results to demonstrate TransMUSE's efficacy. Section 5 discusses the most relevant related work and Section 6 concludes the paper.

2. Problem Formulation

Our aim is to address the challenges of handling spatial heterogeneity of service traffic in edge networks and reducing model training costs when forecasting future demands in edge networks, to support the effective management of their resources.

Formally, multi-service traffic forecasting seeks to maximize the probability that, given T previous measurements of the traffic volume consumed by K services, the predicted traffic consumption over F future time steps is as close as possible to the ground truth. Denoting by x_t^k the traffic volume of service k at timestamp t and $\mathcal{X}_t := [x_t^1, ..., x_t^K]$ the snapshot of all K services at time t, and considering a forecasting model that is parameterized by θ, the multi-service traffic forecasting problem is equivalent to:

$$\arg\max_{\theta} \; p_\theta \left(\mathcal{X}_{t+1}, ..., \mathcal{X}_{t+F} | \mathcal{X}_{t-T+1}, ..., \mathcal{X}_t \right).$$

To solve this problem, we design a Transformer-based Multi-service Traffic Prediction Network (TMTPN) that captures temporal correlations among traffic time series via multi-head attention, then improve forecasting accuracy via service-clustering, as we detail next.

3. TransMUSE Framework

We propose TransMUSE, a deep learning framework for accurate and cost-effective multi-service forecasting at the network edge. Figure 2 gives an overview of the different components this framework entails and the relationship between them, namely:

1. **Edge Node Clustering**: We cluster edge nodes by a set of service-level statistical features using the K-means algorithm, to determine the neural model transfer scope.
2. **Reference Node Selection:** Within each scope, we select the node with the overall highest traffic consumption as the reference node; reference neural models for forecasting will be trained with data collected at such nodes.
3. **Service Grouping:** As certain mobile services exhibit statistical similarities, we cluster services using a modified K-means algorithm based on Wasserstein distance, aiming to reduce the number of multi-service neural models to be employed for prediction.
4. **Model Training:** At the level of each reference node, we train a dedicated TMTPN model for each service cluster, which will simultaneously predict the volume of traffic for all services within such clusters.
5. **Model Transfer:** We transfer the trained reference models from each reference edge node to all other nodes within the corresponding clusters, where they will be applied for inference without further training.

Next, we discuss in details the key stages of our TransMUSE framework.

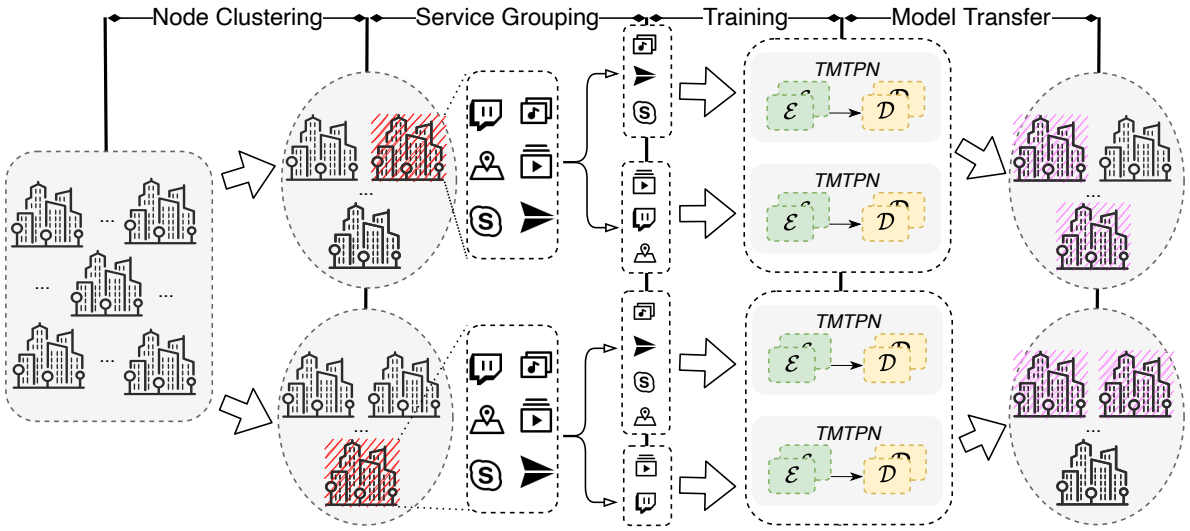


Figure 2: The proposed TransMUSE framework incorporates five stages: 1) Edge nodes are grouped into several *node clusters*, within which model transfer is to be conducted; 2) In each cluster, the edge node with the largest traffic volume is selected as reference (highlighted with hashed patterns); 3) At each reference node, services are further partitioned into *service clusters* by WK-Means; 4) One TMPTN is trained for each service cluster; 5) The models trained on *reference nodes* are transferred to the recipients (highlighted on the right) within the corresponding node clusters.

3.1. Edge Model Transfer

In Multi-access Edge Computing (MEC) scenarios, it is often impractical to train a neural model at each individual edge node, as their computational power is limited and the operational costs and energy expenditure can become prohibitive to operators when deployment density increases. Edge model transfer aims to reduce the cost of measurement collection and model training, by confining these tasks to designated nodes and reusing models trained there on other nodes, without further local tuning. Different from cloud–edge approaches where a central node maintains a global model refined through model updates resulting from local training (federated learning), edge model transfer only considers the model to be transferred among edge nodes without the need for a central cloud. This brings additional merits in terms of data privacy and communication overhead reduction, as the transfer process is confined within a limited scope. A model to be transferred is called a *reference model*, the node where a reference model is trained is called a *reference node*, and the edge nodes that adopt it are referred to as *recipients*.

There are two key issues to address in the edge model transfer process. The first, is determining the scope of model transfer. Different edge nodes may observe distinct traffic patterns due to geographic dissimilarities in terms of mobile user demographics [9] or socioeconomic function (residential areas, business districts, shopping centers, etc.). Edge model transfer, therefore, should be applied across edge nodes (within a cluster) with similar traffic features. Secondly, choosing at which edge node to train a reference model to be transferred within the corresponding cluster will impact the inference accuracy. We put forward an edge model transfer strategy that deals with these two issues as follows:

- **Determining Transfer Scope**: We use K-means clustering to group edge nodes according to nine statistical features, i.e., *mean*, *standard deviation*, *maximum value*, *minimum value*, *skewness*, *kurtosis*, and *the 1st, 2nd, and 3rd quartiles* of traffic volume for each service over the historical traffic data for model training. Specifically, we used an 8:1:1 data split for training, validation and testing. With 20 services, each edge node is represented by a matrix of shape 20×9. A model will only be transferred within the same cluster of edge nodes. In the deployment process in a real network environment, the operator will have the freedom to select a broader range of relevant statistical features for this task.
- **Reference Model Training**: Within each cluster, a reference model will be trained only with data from the edge node where the overall highest traffic consumption is observed. Reference models are then transferred to the recipients within the corresponding clusters. The results we present in Section 4 confirm the generalization abilities of this approach.

At the level of a reference node, a set of neural models will be trained, each of which targets future traffic predictions for groups of services with similar characteristics, as we explain next.

3.2. WK-means Service Clustering

Traffic patterns and volumes may differ among services due to content popularity, number of service subscribers, service scope, etc. (see Figure 1), leading to high information

entropy if observing all services together. Note however that it would be impractical and expensive to maintain a specific prediction model for each service. Therefore, we propose a service clustering algorithm based on the Wasserstein distance (WD) between per-service time-series data points, which we name *WK-means*. We resort to WK-means to reduce the number of prediction models by grouping services into clusters according to their traffic pattern similarity. In such manner, we employ a single model for each service cluster, which can learn specific patterns corresponding to the unique cluster features. Accordingly, we don't have to maintain too many models, thereby reducing training cost and potentially saving energy. By grouping services with WK-means, we ensure the model learns from enough data and a rich set of features, so that it does not perform poorly when predicting on the test set (i.e. we avoid under-fitting).

There are two key factors to consider when measuring time-series similarity, namely, magnitude and 'shape'. The former indicates how comparable the traffic volume of different services is; the latter indicates any similarities in terms of periodicity and short-term temporal patterns. The WD takes these two factor into consideration at the same time, which makes it particularly suitable for our grouping task. Originally the WD was proposed to measure the similarity between two probability distributions, and was recently employed in optimal transportation problems [10]:

$$W_p(\mathcal{P},\mathcal{Q}) = \left(\inf_{\mu\in\Gamma(\mathcal{P},\mathcal{Q})}\int \rho(x,y)^p d\mu(x,y)\right)^{1/p}, \quad (1)$$

where $\mathcal{P}$ and $\mathcal{Q}$ are two probability distributions in $\mathbb{R}^d$, and $\Gamma(\mathcal{P},\mathcal{Q})$ is the set of all probability measures on $\mathbb{R}^d\times\mathbb{R}^d$ with marginals $\mathcal{P}$ and $\mathcal{Q}$. $\rho(x,y)^p$ is a measure of distance between $x\in\mathcal{P}$ and $y\in\mathcal{Q}$ (e.g., $p=2$ for Earth mover's distance). Intuitively, the WD represents the minimal distance for moving the mass of distribution $\mathcal{P}$ to exactly fit the mass of distribution $\mathcal{Q}$.

Unlike other distance metrics, such as Euclidean distance, JensenâĂŞShannon (JS) divergence or KullbackâĂŞLeibler (KL) divergence, WD has the following key advantages: (i) if the target distributions lie in low-dimensional manifolds or share disjoint support, which is not uncommon for high-dimensional data, WD offers a more informative measure (which is not the case for KL and JS divergence that return a constant value or infinity) [11, 12]; and (ii) WD maintains the underlying geometry of the space [13], that is, it not only takes the quantitative value into consideration, but also pays attention to the similarity of distributions' shapes. In contrast, the Euclidean distance cannot quantify shape differences or capture the degree of changes between two times series [14].

Based on WD, we propose the WK-means service clustering algorithm, summarized by the pseudo-code in Algorithm 1. To generate N clusters from S services, WK-means initially sorts all the services by their volume and splits the sorted sequence at $\left[\frac{S}{N}, \frac{2S}{N}, ..., \frac{(N-1)S}{N}\right]$. That is, the sorted sequence is evenly divided into N segments (lines 2-6). WK-means further chooses the service in the middle of each segment as cluster center (lines 8-13). Instead of using random initialization, this approach speeds up the convergence process. Then, the WD between each service and sub-cluster center is calculated, and each service is re-assigned to its nearest sub-cluster (lines 14-24). Finally, each sub-cluster center is updated (line 25) and the previous two steps are iterated until all sub-clusters convergence or the iteration epoch reaches a predefined limit (lines 7-26).

In a setting with multiple edge nodes, it is possible that WK-means may generate different service clustering results on different nodes. To comply with our model transfer strategy, we first conduct WK-means at the level of every edge-node and select the most frequent clustering pattern as the global service grouping. Such pattern is applied to all other nodes in our design.

Algorithm 1 WK-means Service Clustering

Require: X(list::service time series), N(int::cluster), I(int::max iteration)
Ensure: L(list::cluster results)

1: $mean\_list = [\,]$, $centers\_list = [\,]$, $counter = 0$
2: **for** $i = 0 \rightarrow len(X)$ **do**
3: $\quad mean\_list.append(mean(X[i]))$
4: **end for**
5: $mean\_list = sorted(mean\_list)$
6: Initialize X into N clusters by $mean\_list[0 : len(X) : len(X)/N]$
7: **repeat**
8: $\quad$ **for** each cluster C **do**
9: $\quad\quad$ **if** $mean(X[i]) == median(C)$ **then**
10: $\quad\quad\quad C_{center} = X[i]$
11: $\quad\quad$ **end if**
12: $\quad\quad centers\_list.append(C_{center})$
13: $\quad$ **end for**
14: $\quad$ **for** $i = 0 \rightarrow len(X)$ **do**
15: $\quad\quad min\_dist = 10000$
16: $\quad\quad flag = 0$
17: $\quad\quad$ **for** $j = 0 \rightarrow N$ **do**
18: $\quad\quad\quad dist_{ij} = WD(X[i], centers\_list[j])$
19: $\quad\quad\quad$ **if** $dist_{ij} <= min\_dist$ **then**
20: $\quad\quad\quad\quad min\_dist = dist_{ij}$ and $flag = j$
21: $\quad\quad\quad$ **end if**
22: $\quad\quad$ **end for**
23: $\quad\quad L[i] = flag$
24: $\quad$ **end for**
25: $\quad$ update each cluster & $counter++$
26: **until** $counter \geq I$ or L is unchanged

3.3. TMTPN Model Design

To perform multi-service traffic forecasting, we design Transformer-based Multi-service Traffic Prediction Networks (TMTPNs), each of which inherits from the canonical Transformer architecture and is dedicated to each individual service cluster. Transformers have shown remarkable performance in processing sequential data and have been

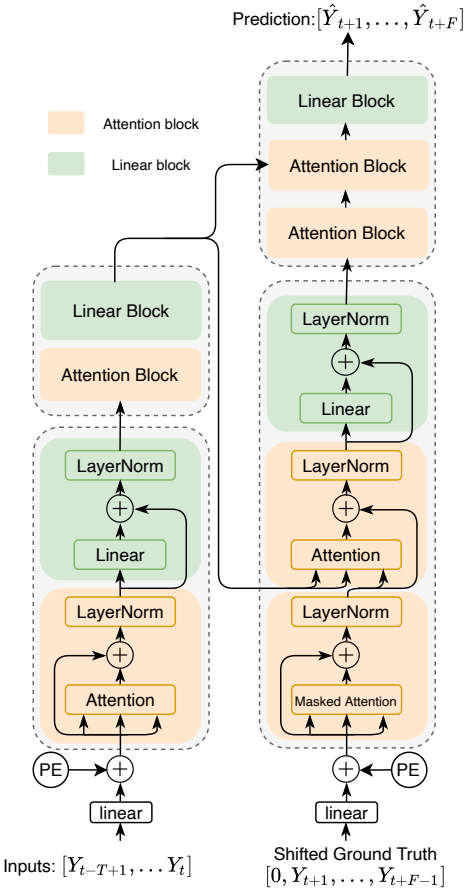


Figure 3: TMTPN architecture based on an Encoder-Decoder design. Historical traffic input combined with positional encoding (PE) is processed by the Encoder, which gives the output to each Decoder block. Within each, the core components are a multi-head Attention block and a Linear block.

adopted previously for natural language processing [15], computer vision [16], and vehicular traffic prediction [17] tasks. Our TMTPN model is illustrated in Figure 3 and follows an Encoder-Decoder paradigm, encompassing the following components:

- **Multi-Head Attention:** Multi-head attention consists of multiple scaled dot-product attention structures that capture temporal dependencies in long sequences. The attention block receives three inputs: $Q \in \mathbb{R}^{T\times d_k}$ (query), $K \in \mathbb{R}^{T\times d_k}$ (key) and $V \in \mathbb{R}^{T\times d_k}$ (value), in which T represents the sequence length and d_k is the embedding dimension of each item in the sequence. Attention is computed as:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

  QK^T generates a $T \times T$ matrix of alignment scores, where each entry denotes the correlations between two instances in the sequence. The matrix is scaled and then multiplied by V to generate the hidden representation of the input that incorporates attention information. Multi-head attention splits Q, K and V into multiple chunks, which are processed with independent attention blocks. The outputs of all the attention blocks are concatenated and projected back into hyperspace $\mathbb{R}^{d_k}$.

- **Encoder and Decoder Layers**: The encoder layers (Figure 3 left) contain a multi-head attention block and a linear block, each of which utilizes a skip connection and layer normalization to prevent over-fitting. For the encoder, only the input sequence is given to the multi-head attention block, i.e., $X = Q = K = V$, where self-attention is computed. The decoder (Figure 3 right) incorporates an extra attention block. Specifically, the first attention block in the decoder computes the self-attentional representations of the decoder input, and the second block takes the encoder output as the key $K \in \mathbb{R}^{T\times d_k}$ and the value $V \in \mathbb{R}^{T\times d_k}$, querying which historical inputs are important when making future predictions.

- **Positional Encoding (PE):** Since transformers do not contain any sequential structure, timing features are not encoded in the network by default. Therefore, positional encoding is added to the input sequence, which reflects the relative position of each timestamp. PE is computed as:

$$PE(pos, 2i) = \sin(pos/10,000^{2i/d_{model}}),$$
$$PE(pos, 2i+1) = \cos(pos/10,000^{2i/d_{model}}),$$

  where pos denotes the position index of the item in the sequence and d_{model} is the dimension of the encoded position.

- **Parallel Decoding:** Traditional seq2seq models [7] perform decoding in an auto-regressive manner during training. That is, decoding the t^{th} element in a sequence relies on the hidden states passed from timestamp $t-1$ and the decoded $(t-1)^{\text{th}}$ item, which are provided as the input. It is therefore impossible to decode all the items in parallel. Transformers overcome this problem during training by introducing the *shifted decoder input* and *look-ahead mask*. Assume that the ground truth to be provided to the decoder is $Y = [y_1, ..., y_F]$, then the input is the shifted-right ground truth $X = [0, y_1, ..., y_{F-1}]$. The *look-ahead mask* (M) is introduced when computing the alignment scores as follows:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}M\right),$$

  where $X = Q = K$, and M is a $F\times F$ matrix with each entry above the diagonal equal to negative infinity, and below/on the diagonal equal to 0. The scaled matrix of alignment scores is masked with M, which yield a $F \times F$ lower triangular matrix, meaning that at a

given timestamp i, there is no correlation ($A_{ij} = 0$) with the input from any future timestamp j ($j > i$). By masking, the decoder can approximate the output at F timestamps, $\hat{Y} = [\hat{y}_1, ..., \hat{y}_F]$, in parallel. This technique is only applied during training, while during testing the transformer decodes step-by-step, as seq2seq models.

Overall, the proposed TMTPN architecture has several merits: (i) it can be trained fast due as the look-ahead mask and the shifted decoder input that facilitate parallelization; (ii) it can process longer sequences than traditional seq2seq models; and (iii) it captures the most essential historical information that impacts most on prediction results, irrespective of the length of an input sequence, thanks to Position Encoding and Multi-Head Attention.

4. Experiments

We implement TransMUSE and its TMTPN models, as well as a set of benchmark neural models in Tensorflow v2.3.0 using the cuDNN v7.6 and CUDA v10.1 libraries. To demonstrate the performance gains of our solution, we train and evaluate the neural models and experiment on a large-scale real-world wired network traffic dataset collected by a network operator in Sichuan Province, China. For this, we employ a high-performance computing cluster comprising 12 servers, each equipped with a 32-cores Intel E5-2620 CPU and running Red Hat Enterprise Linux, and accelerate the training process with multiple GPUs out of a pool of 96 Nvidia RT2080Ti units.

We conduct three sets of experiments to demonstrate (1) multi-service traffic prediction performance gains attained by our TMTPN models; (2) the benefit of employing service clustering with WK-means; and (3) forecasting performance with edge model transfer.

4.1. Dataset & Pre-processing

The dataset we employ was collected in a city with over 6 km^2 land coverage, administratively divided into 7 districts and 1 core urban area, and with a population of approximately 2.7 million inhabitants. The traffic within each district (D1 to D8) is handled by a dedicated edge node, and the high level structure of the deployment is illustrated in Figure 4. Traffic data was collected by Deep Packet Inspection (DPI) via port mirroring, between July 1st and 31st, 2020. Traffic was aggregated at session level, with only application type, district identifier, direction (uplink/downlink), total volume and timing information (session start/end) being recorded, to preserved anonymity. In total, 20 service types are distinguished, as summarised in Table 1, where these are sorted in descending order by their volume across the entire deployment, and indexed. The traffic volume distribution for the top-8 services is shown in Figure 5, where bars depict the fraction of the overall volume and the line the corresponding values. Due to the commercially sensitive nature of the dataset, we cannot disclose the precise identity

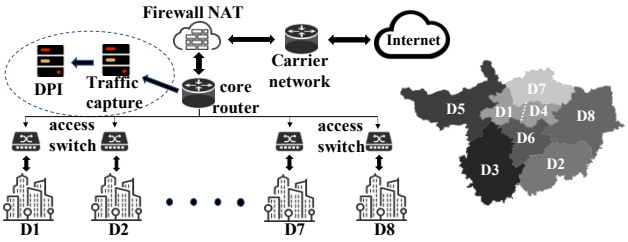


Figure 4: Network topology and district map of the target city. Traffic measurements collected via DPI and further processed at the core router.

of the city, nor the specific service names for which traffic measurements were collected.

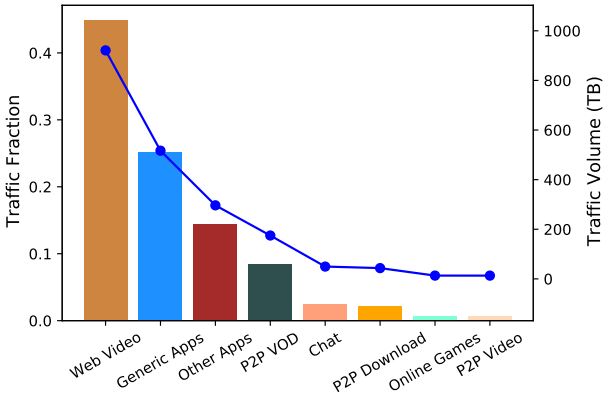


Figure 5: Service traffic volume (line) and fraction of the total (bar) in the city across 8 districts over 31 days.

Over the 31 consecutive days of measurements, we sample the traffic consumption every minute, assuming uniform consumption per session throughout their duration. This is reasonable, given the predominantly short-lived nature of sessions, leading to temporal sequences of 44,640 data points for each service in each region. We normalize service traffic volumes to the 0–1 range, to ensure similar magnitudes during training. We use an 80/10/10 data split for training, validation, and testing, and train models separately on a region-by-region basis.

4.2. Benchmarks & Metrics

For comparison, we consider the following state-of-the-art DL models as baselines:

- **LSTM**, which is now a classic structure for tackling regression tasks, and has been extensively used for traffic prediction [18, 19]. We implement a three-layer LSTM, which offers an appropriate complexity–effectiveness tradeoff.

1: Web Video	**2:** Generic Apps	**3:** Other Apps	**4:** P2P VOD	**5:** Chat	**6:** P2P Download	**7:** Online Games
8: P2P Video	**9:** Cloud Storage	**10:** Shopping Online	**11:** Live Stream	**12:** Music Stream	**13:** News Apps	**14:** Generic Video
15: VoIP	**16:** Stock Apps	**17:** Travelling Websites	**18:** Mail Apps	**19:** Living Apps	**20:** Portal Webs	

Table 1
Service names and indexing for the traffic dataset used in our experiments.

Model\District	D1		D2		D3		D4		D5		D6		D7		D8	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
LSTM	45.20	125.90	47.05	136.75	51.37	145.39	21.20	60.98	21.65	63.74	15.69	44.36	18.78	56.67	50.43	147.15
AttentionAR	61.80	176.89	63.08	184.99	65.40	181.71	24.24	66.58	27.79	81.35	19.06	53.05	23.70	70.21	64.11	180.17
GraphConv	107.42	300.25	110.15	327.21	116.61	317.44	36.44	99.49	40.93	123.34	23.18	66.13	34.41	108.22	118.35	340.68
MTNet	44.15	127.37	46.78	136.54	48.97	138.71	21.86	62.33	23.70	71.22	16.40	48.22	18.54	57.21	46.51	133.26
GASTN	48.28	137.85	47.91	145.03	53.92	156.82	20.98	58.25	23.30	72.51	15.83	44.88	20.47	61.13	51.29	149.57
TMTPN (ours)	**37.69**	**106.19**	**38.16**	**118.76**	**43.27**	**124.86**	**16.57**	**46.80**	**17.99**	**53.91**	**12.04**	**34.94**	**14.91**	**45.98**	**41.49**	**117.44**

Table 2
MAE and RMSE performance (in MB) on the multi-service traffic forecasting task with LSTM, AttentionAR, GraphConv, MTNet, GASTN and our TMTPN across 8 districts.

- **MTNet**, which was designed for multivariate time series prediction and adopts an encoder-decoder architecture to extract both long- and short-term hidden representations correlation among these [2].
- **GraphConv**, which is also aimed at tackling multiple time series predictions [20], integrating graph convolution [21] and an LSTM network to extract correlations between multiple sequences and temporal patterns. We employ the Spektral library for our implementation [22].
- **AttentionAR**, a model that we implement based on the Bahdanau Attention structure [23] with rolling prediction through a LSTM cell. Attention is used to assign weights to historical input.
- **GASTN**, which was proposed in [24] for mobile traffic prediction based on attention and recurrent neural networks.

To evaluate the performance of our proposed models and that of the benchmarks considered, we compute the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). In essence, these metrics quantify the difference between ground-truth and predicted values, and are defined as follows [25]:

$$MAE = \frac{1}{S \times F} \sum_{i=1}^{S} \sum_{j=1}^{F} |y_{ij} - \hat{y}_{ij}|; \tag{2}$$

$$RMSE = \sqrt{\frac{1}{S \times F} \sum_{i=1}^{S} \sum_{j=1}^{F} (y_{ij} - \hat{y}_{ij})^2}, \tag{3}$$

where S is the number of services, T represents the number of prediction steps, and y_{ij} and $\hat{y_{ij}}$ denote the ground truth and respectively predicted traffic volume for service i at timestamp j.

4.3. Multi-service Traffic Prediction by TMTPN

We first examine TMTPN's performance vis-a-vis that of the benchmarks considered, then investigate the influence that input/output lengths have on this.

4.3.1. Forecasting Comparison

We first train and test different models for every district separately, using traffic solely observed within each of these. We take input sequences of length 30 (i.e., 30-min historical data) and predict the traffic volume per service over 5 future timestamps. The obtained results are summarized in Table 2, where lower MAE and RMSE values indicate superior prediction performance.

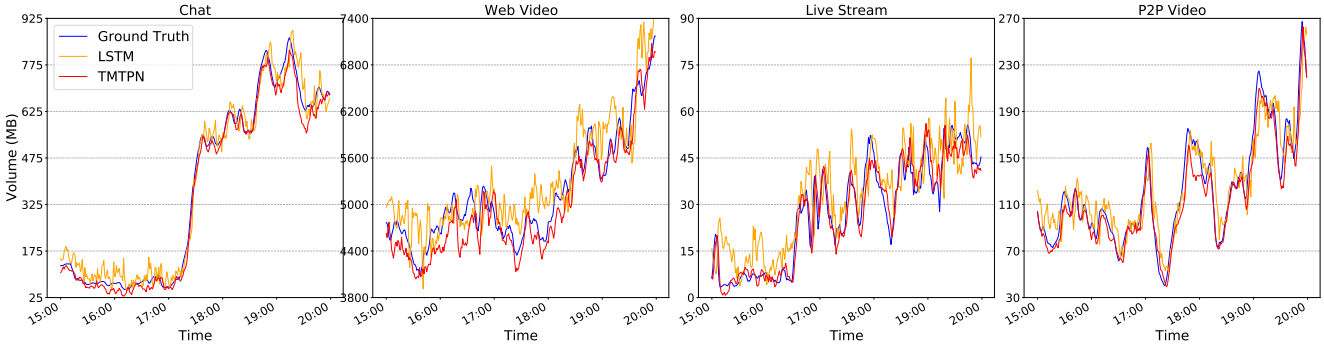
Observe that the TMTPN models we propose consistently outperform the state-of-the-art neural networks considered. In particular, when compared with the second best model, LSTM, our TMTPN reduces the MAE and RMSE on avergae by 18.95% and respectively 17.74%, across the eight districts. This is because the multi-head attention structure adopted by our design allows the model to jointly extract information from different representation sub-spaces at different points in time, giving higher weights to the most significant historical patterns, to enhance prediction performance.

The performance of LSTM and MTNet is relatively similar. GASTN's performance is inferior to that of MTNet in districts with large traffic volumes, but outperforms MTNet in districts where the traffic volume is smaller, i.e., D4, D5, D6. However, AttentionAR largely overestimates traffic demand, indicating that the traditional attention mechanism is not best-suited to multi-service prediction tasks. Finally, GraphConv performs poorly in comparison with our TMTPN and the other two benchmarks we consider. This is likely because no strong spatial relationships between the different services exist in edge network settings.

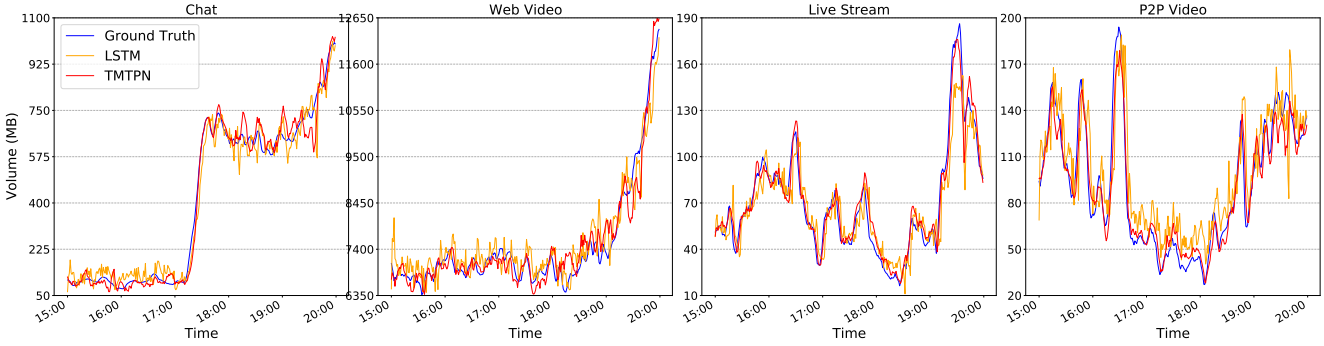
To better appreciate the forecasting performance of our proposed TMTPN model at service level, in Table 3 we summarize that across two districts with dissimilar service

DIS	Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
D1	**TMTPN**	**187.05**	**250.79**	**97.91**	**52.54**	**23.71**	**31.94**	**5.36**	**10.85**	**16.71**	**24.90**	**4.32**	11.01	**2.77**	**1.13**	**0.59**	**0.58**	0.29	0.12	0.12	**0.03**
	LSTM	257.40	310.18	121.12	65.28	33.13	49.63	7.92	15.47	17.39	36.40	6.26	**10.35**	2.81	2.00	0.82	0.70	**0.26**	**0.10**	**0.09**	0.03
D2	**TMTPN**	**270.20**	**241.52**	**113.04**	**56.58**	**29.85**	**34.36**	**6.46**	**10.15**	**17.50**	**8.83**	**5.66**	9.30	**2.53**	**1.04**	**0.53**	**0.43**	**0.08**	0.07	0.13	**0.03**
	LSTM	277.28	293.57	125.73	68.20	30.70	47.22	8.12	15.04	18.88	11.03	7.09	**8.73**	2.65	1.49	0.67	0.45	0.08	**0.06**	**0.12**	0.03

Table 3
Per-service prediction performance in terms of MAE (MB) in districts D1 and D2. Service names given in Table 1 by index.



(a) Service-level predictions in district D1



(b) Service-level predictions in district D2

Figure 6: 5-step forecasting performance with TMTPN and LSTM at service level over 5 busy hours (15:00 to 20:00 on 29 July, 2020) vs Ground truth.

usage patterns, namely D1 and D2, while in Figure 6 we illustrate 5-step prediction instances performed in the two districts across 5-hour windows (busy hours between 15:00 and 20:00 on 29 July, 2020) for 4 randomly selected services (chat, web video, live streaming, P2P video). We only compare TMTPN with LSTM in this and the subsequent experiments, because LSTM appears to be an effective deep learning model that achieves solid performance, which the results in Table 2 and prior work [6, 7] confirm. As can been seen from the figure, TMTPN is superior to LSTM, as it tracks more closely the ground truth traffic that would be available under ideal circumstances. This is especially clear to observe on 'Chat' traffic forecasting in D1 (sub-figure (a)) and 'Live Streaming' traffic in D2 (sub-figure (b)).

4.3.2. Impact of Input and Output Length

In this subsection, we evaluate the long-term and short-term prediction performance of our TMTPN, focusing again on districts D1 and D2. We examine the MAE across all services as the forecasting horizon varies between 5 and 30 steps, while we also vary the input size, i.e., 5, 15 and 30 historical traffic snapshots. The obtained results are summarized in Figure 7, where the x-axis represents the combination of input and output length (e.g., 30-5 indicates the model uses the previous 30 minutes traffic data to predict the upcoming 5 minutes traffic demand). The top sub-figure corresponds to district D1 and the bottom to district D2.

Observe that TMTPN is consistently superior to LSTM, as it achieves lower prediction errors. The performance gains grow with the length of the forecasting horizon, with TMTPN reducing the MAE experienced with LSTM on long-term predictions by 43.21% and 40.77% in district D1 and D2, respectively. Benefits are also observable short-term, where TMTPN attains 15.02% and respectively 22.74% lower MAE than LSTM in the two districts, when the input and output lengths are both 5 (5-5). These gains can be attributed to the multi-head attention mechanism that our design adopts. In addition, the shifted input with look-ahead mask not only enables training parallelization, but also

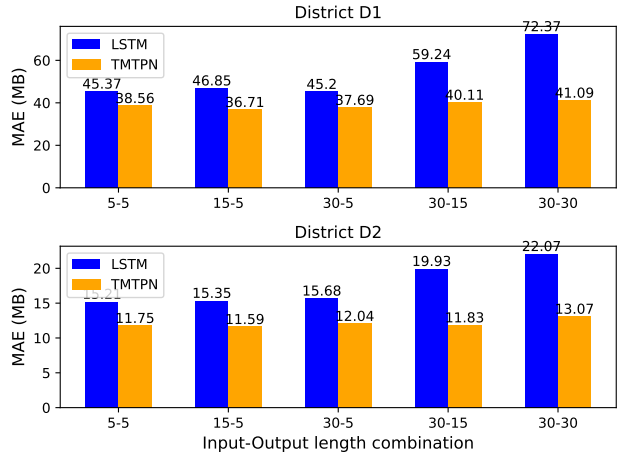


Figure 7: Impact of different Input-Output lengths on forecasting performance (in terms of MAE) with LSTM and our TMTPN in districts D1 (top) and D2 (bottom).

ensures TMTPN can predict the future sequence on a rolling basis, unlike the LSTM, which predicts multiple future steps at once and is thus prone to larger errors.

Lastly, we note that the input length has only marginal impact on TMTPN's forecasting accuracy, with input size impacting performance slightly differently at the level of the two districts examined. Yet in both cases the best performance is attained with 15 historical snapshots. Based on these results, we argue that if the input length is too short (5), the model may not be able to capture certain periodic information or longer trends.

4.4. Multi-service Clustering

Recall that the aim of service clustering in TransMUSE is to further improve forecasting performance by grouping services into different clusters, according to their temporal similarity. Here, we demonstrate the benefits of using our WK-means algorithm for this task (hereafter denoted as WASS), as compared to three benchmarks that can be applied to time series data, namely K-means clustering based on Euclidean distance (EUC), K-means based on Cosine similarity [26] (COS), and (3) Derivative Dynamic Time Warping (DTW) clustering [6] implemented with the `tslearn` library [27].

4.4.1. Number of Clusters

Before comparing the clustering algorithms, the appropriate number of clusters K needs to be determined. The Silhouette score is routinely employed to characterize clustering performance, which is computed as the difference between the mean of the intra-cluster distances and the mean of the nearest-cluster distances, normalized by the maximum between the two [28]. The silhouette score ranges between [-1, 1], where 1 indicates the best cluster separation, values near 0 indicate overlapping clusters, and negative ones suggest that a sample has been assigned to the wrong cluster.

K\Algorithm	EUC	COS	DTW	WASS
2	**0.851**	**0.157**	**0.852**	**0.869**
3	0.788	0.06	0.814	0.779
4	0.706	-0.09	0.782	0.801
5	0.651	-0.109	0.712	0.732

Table 4
Silhouette score comparison with associated distance metric for different numbers of clusters K and the four clustering algorithms considered, in district D1.

With this, we validate the effectiveness of our WK-means algorithm vis-a-vis that of the benchmarks considered, on the eight districts separately. For each district, K is chosen in the $\{2, \ldots, 5\}$ range, and we compute the silhouette score for each K value. The results on district D1 are given in Table 4, which suggest $K = 2$ is the optimum value. The same holds for the vast majority of other districts, with $K = 3$ yielding marginally higher silhouette scores (0.01 difference) in 2 out of 32 instances. Hence we select $K = 2$ for all the remaining experiments. When the number of services grows, the same approach is suitable.

4.4.2. Clustering Algorithms Comparison

Next we evaluate forecasting performance with TMTPN when a model is trained individually on services clusters, following grouping of the 20 services into $K = 2$ clusters using the proposed WK-means and the benchmark algorithms. We resort again to MAE and RMSE for evaluation and summarize the results obtained in Table 5.

The results demonstrate that all clustering algorithms can reduce the prediction errors, which is more apparent in districts with larger traffic volumes, such as D1, D2, D3 and D8. Our WASS solution is superior to COS, because cosine similarity gives priority to the direction of two vectors, such as the semantic similarity between two sentences. DTW and EUC are essentially based on the Euclidean distance between traffic magnitude, whereas the "shape" of a time series is an important feature when measuring the similarity between two time series. Our WK-means algorithm (WASS) based on Wasserstein distance possesses such ability, which is reflected in the lower prediction errors obtained (bottom row in Table 5).

4.4.3. Cluster Visualization

To better appreciate where the differences in the performance attained with WK-means and the 3 benchmarks stem from, in Figure 8, we visualize the cluster membership of the different services in a randomly chosen district (D2). We can draw the conclusion that the traffic magnitude and 'shape' do have an impact on the clustering results. Observe that both DTW and EUC rely on the Euclidean distance and only services with very large traffic magnitude are grouped into the same cluster. The fact that service No. 2 belongs to different clusters in DTW and EUC confirms the importance

Algorithm\District	D1		D2		D3		D4		D5		D6		D7		D8	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
No Custering	37.69	106.19	38.16	118.76	43.26	124.86	16.57	46.80	17.99	53.91	12.04	34.94	14.91	45.98	41.49	117.43
EUC	31.22	81.26	31.68	86.91	33.67	88.83	15.00	42.63	16.49	47.62	10.80	30.77	13.21	39.82	33.04	91.57
COS	35.78	100.92	37.18	109.03	41.08	116.10	15.67	43.47	17.83	52.98	11.74	34.19	14.46	43.59	38.65	111.33
DTW	31.22	81.26	33.35	92.38	35.92	98.38	15.45	42.49	16.49	47.62	10.80	30.77	13.21	39.82	34.82	92.63
WASS(ours)	**28.98**	**81.14**	**30.60**	**85.09**	**32.20**	**72.87**	**14.65**	**39.36**	**15.76**	**46.91**	**10.45**	**30.37**	**12.99**	**39.14**	**32.32**	**87.95**

Table 5
Clustering algorithms comparison based on MAE and RMSE (MB) of forecasts obtained with TMTPN applied on service clusters across the 8 districts.

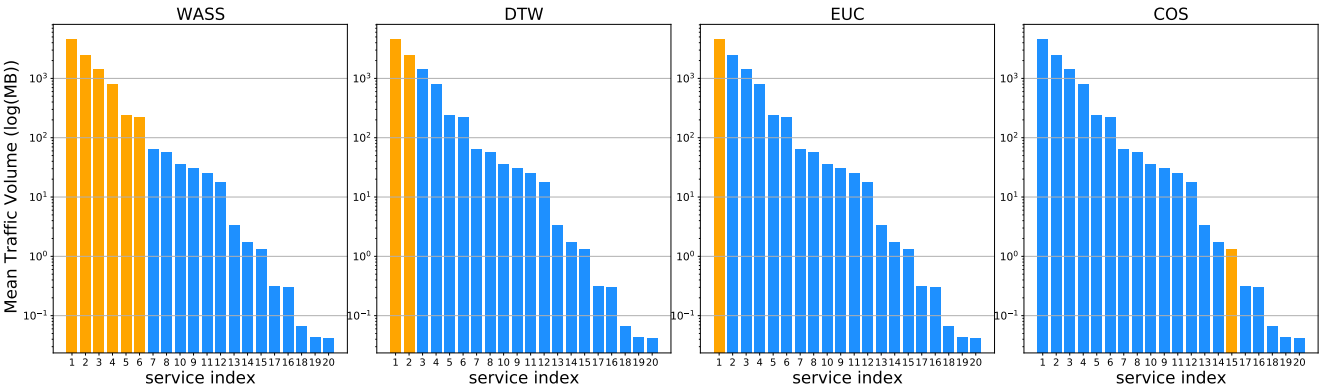


Figure 8: Clustering visualization for the four algorithms. Service bars with the same colour are grouped in the same cluster. Service indexes are sorted in descending by traffic volume, and the service name can be obtained by mapping in Table 1

of service traffic shape. In contrast, WASS clusters services from 1 to 6 into one category even though their traffic quantities are distinct. Figure 9 further illustrates the importance of traffic 'shape' as captured with our WASS approach. When we compare cluster 1 and cluster 2 in each sub-figure, it is obvious that the traffic volume range is different. Observe the sub-figures corresponding to WASS, where the services in each cluster share similar traffic patterns, albeit having services with different traffic volume in cluster 1. In contrast, the service shapes in cluster 2 obtained with DTW are more heterogeneous. Cosine similarity pays more attention to the difference between two vectors in direction rather than distance or length. In our service clustering task, the traffic magnitude is the primary consideration, therefore cosine similarity is less effective in clustering service time series, which is also confirmed by our previous results reported in Table 5, where COS performs worst than the other three algorithms in all 8 districts.

4.5. Edge Model Transfer

Finally, we demonstrate the merits of model transfer in TransMUSE by showing that reusing models trained at reference nodes within a node cluster, with the aim of reducing computational overhead, does not impact negatively on the forecasting performance.

4.5.1. Region Clustering

Recall that the first step in transferring reference models is to decide the transfer scope. We use K-means clustering to group regions according to the statistical features of all the service traffic time series. We resort again to the silhouette score to determine the optimal number of region clusters, which we compute for $k \in \{2, 3, 4, 5, 6, 7\}$ in Table 6. We conclude that $k = 2$ produces the highest score and districts D1, D2, D3, D5, and D8 should be grouped together, with the remaining 3 regions belonging to the second edge node cluster.

k	2	3	4	5	6	7
score	**0.441**	0.380	0.341	0.259	0.112	0.109

Table 6
Silhouette score of edge node clustering by K-means based on 9 statistical features with different number of clusters k from 2 to 7.

Our focus is on maintaining a small number of models while ensuring model transferability. Our real-world dataset contains district-level edge nodes (rather than base stations). In deployments with many more edge nodes, one can put constraints on the maximum number of groups/the maximum number of members in a group, to maintain a desired complexity/transferability trade-off.

4.5.2. Model Transfer Validation

We order the regions according to the overall traffic volume and conclude that D4 and D3 are to be selected as reference nodes for cluster 1 and 2, respectively. We quantify the generalization ability of the models trained by comparing

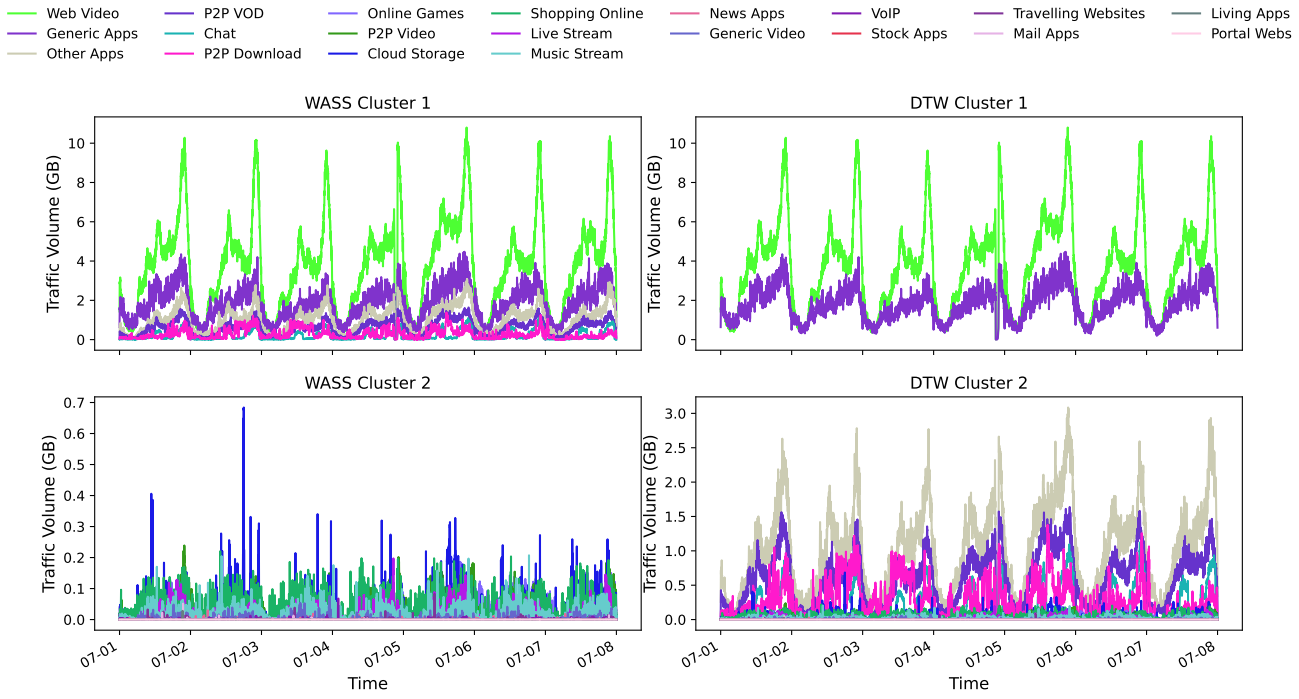


Figure 9: Service cluster membership and traffic pattern visualization over one week in District No. 2 (D2) when using WASS (left) and DTW (right).

the RMSE when performing forecasting following model transfer (TransMUSE) versus when models are trained locally at individual region level (Original). To add further perspective and verify our hypothesis that models trained at edge nodes witnessing large traffic volumes have stronger generalization abilities, we also examine the forecasting performance when models are trained on regions where the traffic volume is the lowest among cluster members, prior to transfer (Ctrl-Exp).

The result are illustrated in Figure 10 for the two clusters, where regions appear in descending order by the overall traffic volume. Observe that when the reference models are trained on regions with the highest traffic demand (TransMUSE), the RMSE values are almost identical to those obtained when training models individually at each edge node. The largest performance gap is at the level of D6, where a 0.26% performance degradation is observed in terms of forecasting accuracy (RMSE). In contrast, if reference models were to be trained at edge nodes with lowest traffic volumes (D6 in cluster 1 and D5 in cluster 2), the forecasting performance would suffer (Ctrl-Exp). Specifically, the averaged RMSE error over 8 regions is 9.26 MB, which is 92 times larger than with TransMUSE.

We conclude that, as the number of edge nodes increases with the growing adoption of the MEC paradigm, our proposed model transfer strategy will help reduce training time and energy consumption. TransMUSE will only need to revisit cluster membership and will circumvent the need to persistently collect taffic data in each district.

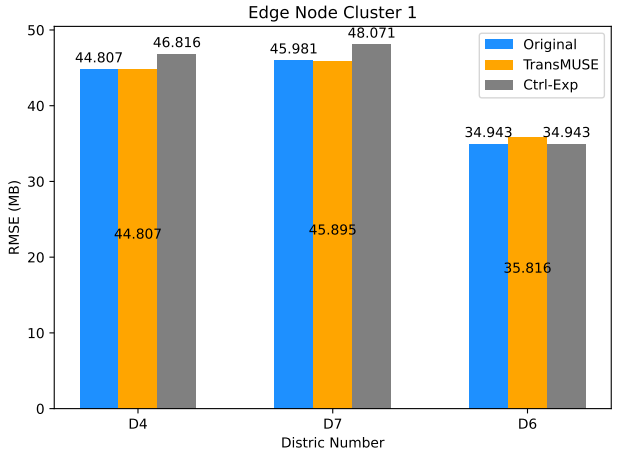
5. Related Work

Network traffic prediction is critical to network resource management, optimization and QoS improvement. While this topic has received a lot of attention over the recent years, aspects including service-level traffic forecasting and predicting with low computational overhead have been largely overlooked. Here, we summarise the most relevant work related to our contribution.

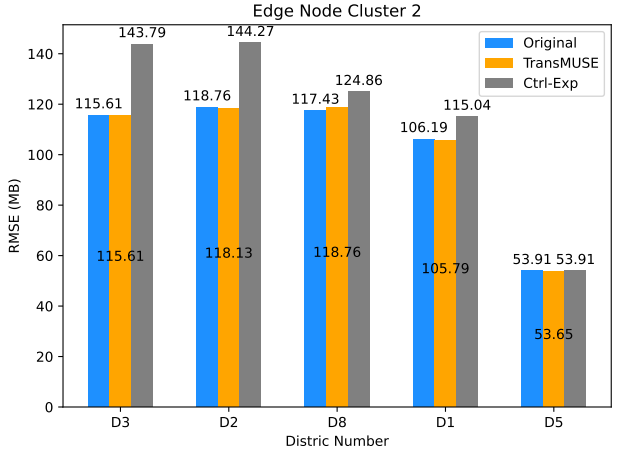
5.1. Traffic Forecasting

The main approaches to time sequence prediction are State Space Models (SSMs) and sequential models that frequently use deep learning (DL) [29]. The most representative SSMs are Auto-Regressive Integrated Moving Average (ARIMA) models and variants of these, which have been widely adopted for mobile traffic forecasting [30, 31, 32]. Their major drawback is that they require manual parameter selection on a sequence-by-sequence basis. In addition, they perform poorly when inputs exhibit high variability.

DL has made advances in multiple domains, with Long-Short Term Memory (LSTM) models proven to be superior to traditional models such as ARIMA when predicting wired and wireless traffic [18, 33, 34, 35]. Given that spatial correlations exist between traffic generated at different base stations in wireless networks, LSTM models have been combined with Convolutional Neural Networks (CNNs) to tackle this problem. Zhang et al. proposed a ConvLSTM model to predict multi-service mobile traffic [7] and a graph-sequence spatio-temporal model is introduced in [36] to



(a) Model transfer performance in Edge node cluster 1. The reference models are trained in district D4 (TransMUSE) with the highest traffic volume and D6, which sees the lowest traffic consumption (Ctrl-Exp).



(b) Transfer performance comparison in Edge node cluster 2. The reference models in district D3 (TransMUSE) with the highest traffic volume and D5, which sees the lowest traffic consumption (Ctrl-Exp).

Figure 10: Traffic forecasting performance comparison when model transfer is employed based on highest traffic demand (TransMUSE), a control experiment where reference models are trained at lightly-loaded edge nodes (Ctrl-Exp), and no edge node clustering is performed, i.e. models trained individually at each location (Original).

forecast cellular traffic demand. More recently, attention and transformer architectures demonstrated the ability to handle long sentences in the NLP domain, which subsequently led to their adoption in time series forecasting tasks [6, 29].

However, none of these prior works builds on the observation that spatial correlations are weak in wired networks and correlations among services matter most.

5.2. Edge Model Transfer

As edge computing is getting traction, there have been several research projects focusing on cloud-edge model training based on collaborative learning. He et al. design a collaborative global-local learning scheme that leverages the generalization capability of the global model and the personalization ability of local models to boost the training performance of a graph attention spatio-temporal network (GASTN) for city-wide mobile traffic prediction [6]. Yan et al. propose COLLA, a collaborative learning framework that allows devices and the cloud to learn collectively user locations [37]. Zhang et al. design a collaborative cloud-edge computation method for driving behavior modeling, which trains and prunes common models in the cloud and conducts transfer learning at the edge [38]. Cartel is proposed in [39] for cloud-edge collaborative learning, aiming at distributing and updating machine learning models across geographically distributed edge clouds.

These works are mostly set on the premise that there exists plenty of data in the cloud to train global models. Edge-edge collaboration, in scenarios where data is largely available only at the network edge, has received less attention. Further, the cost of data transfer overhead has been thus far overlooked, which is non-negligible for network operators.

6. Conclusion

In this paper, we tackled network traffic prediction in multi-service edge networks with spatially heterogeneous demands. We proposed TransMUSE, a framework that groups edge nodes into cohorts and trains transformer-based (TMTPN) models at reference locations, which can be transferred within cohorts without any adaptation. By means of extensive experiments with real-world data, we demonstrated TransMUSE's forecasting performance is comparable with that of training individual models with local data at each node. We further propose WK-means, a service clustering routine, which allows to reduce the number of TMTPN models to be maintained for forecasting, based on service similarities. All of these facilitate accurate short- and long-term multi-service traffic prediction with reduced measurement and training costs, which is essential for fine-grained network management.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (Grant No.U1909204), the National Key R&D Program of China (Grant No.2020YFB1-806002), Beijing Natural Science Foundation, China (No. 4202082), the China Scholarship Council and Youth Innovation Promotion Association of Chinese Academy of Sciences (2021168).

References

[1] Y. Fang, A. Diallo, C. Zhang, P. Patras, Spider: Deep learning-driven sparse mobile traffic measurement collection and reconstruction, in: 2021 IEEE Global Communications Conference, Institute of Electrical and Electronics Engineers (IEEE), 2021.

[2] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, S.-D. Lin, A memory-network based solution for multivariate time-series forecasting, arXiv preprint arXiv:1809.02105 (2018).

[3] C.-W. Huang, C.-T. Chiang, Q. Li, A study of deep learning networks on mobile traffic forecasting, in: IEEE PIMRC, 2017.

[4] C. Zhang, P. Patras, Long-term mobile traffic forecasting using deep spatio-temporal neural networks, in: ACM MobiHoc, 2018, pp. 231–240.

[5] X. Wang, Z. Zhou, F. Xiao, K. Xing, Z. Yang, Y. Liu, C. Peng, Spatio-temporal analysis and prediction of cellular traffic in metropolis, IEEE Transactions on Mobile Computing 18 (9) (2018) 2190–2202.

[6] K. He, X. Chen, Q. Wu, S. Yu, Z. Zhou, Graph attention spatial-temporal network with collaborative global-local learning for city-wide mobile traffic prediction, IEEE Transactions on Mobile Computing (2020).

[7] C. Zhang, M. Fiore, P. Patras, Multi-service mobile traffic forecasting via convolutional long short-term memories, in: IEEE International Symposium on Measurements & Networking (M&N), 2019.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[9] R. Singh, M. Fiore, M. Marina, A. Tarable, A. Nordio, Urban Vibes and Rural Charms: Analysis of Geographic Diversity in Mobile Service Usage at National Scale, in: The World Wide Web Conference, 2019, pp. 1724–1734.

[10] J. Shen, Y. Qu, W. Zhang, Y. Yu, Wasserstein distance guided representation learning for domain adaptation, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[11] H. Liu, X. Gu, D. Samaras, Wasserstein GAN with quadratic transport cost, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 4832–4841.

[12] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International conference on machine learning, PMLR, 2017, pp. 214–223.

[13] L. Wasserman, Optimal transport and Wasserstein distance, [EB/OL], http://www.stat.cmu.edu/~larry/=sml/Opt.pdf (2020).

[14] X.-L. Dong, C.-K. Gu, Z.-O. Wang, Research on shape-based time series similarity measure, in: International Conference on Machine Learning and Cybernetics, IEEE, 2006, pp. 1253–1258.

[15] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[16] X. Pan, Z. Xia, S. Song, L. E. Li, G. Huang, 3D object detection with pointformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7463–7472.

[17] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, H. Xiong, Spatial-temporal transformer networks for traffic flow forecasting, arXiv preprint arXiv:2001.02908 (2020).

[18] H. D. Trinh, L. Giupponi, P. Dini, Mobile traffic prediction from raw data using LSTM networks, in: International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), IEEE, 2018, pp. 1827–1832.

[19] R. Vinayakumar, K. Soman, P. Poornachandran, Applying deep learning approaches for network traffic prediction, in: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017, pp. 2353–2358.

[20] M. Cerliani, Time Series Forecasting with Graph Convolutional Neural Network, [EB/OL], https://towardsdatascience.com/time-series-forecasting-with-graph-convolutional-neural-network-7ffb3b70afcf (2020).

[21] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[22] D. Grattarola, Spektral for Graph Deep Learning, [EB/OL], https://github.com/danielegrattarola/spektral (2021).

[23] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[24] Q. Wu, K. He, X. Chen, S. Yu, J. Zhang, Deep transfer learning across cities for mobile traffic prediction, IEEE/ACM Transactions on Networking (2021).

[25] M. M. Krell, B. Wehbe, A first step towards distribution invariant regression metrics, arXiv preprint arXiv:2009.05176 (2020).

[26] X. Ren, H. Gu, W. Wei, Tree-RNN: Tree structural recurrent neural network for network traffic classification, Expert Systems with Applications 167 (2021) 114363.

[27] Tslearn for the analysis of time series, [EB/OL], https://tslearn.readthedocs.io/en/stable/ (2017).

[28] K. R. Shahapure, C. Nicholas, Cluster quality analysis using silhouette score, in: 7th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2020, pp. 747–748.

[29] N. Wu, B. Green, X. Ben, S. O'Banion, Deep transformer models for time series forecasting: The influenza prevalence case, arXiv preprint arXiv:2001.08317 (2020).

[30] A. Adas, Traffic models in broadband networks, IEEE communications Magazine 35 (7) (1997) 82–89.

[31] Y. Shu, M. Yu, O. Yang, J. Liu, H. Feng, Wireless traffic modeling and prediction using seasonal ARIMA models, IEICE transactions on communications 88 (10) (2005) 3992–3999.

[32] K. Sultan, H. Ali, Z. Zhang, Call detail records driven anomaly detection and traffic prediction in mobile cellular networks, IEEE Access 6 (2018) 41728–41737.

[33] A. Rago, G. Piro, G. Boggia, P. Dini, Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction, IEEE Transactions on Vehicular Technology 69 (9) (2020) 10362–10374.

[34] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal 5 (1) (2017) 450–465.

[35] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, H. Zhang, Deep learning with long short-term memory for time series prediction, IEEE Communications Magazine 57 (6) (2019) 114–119.

[36] L. Fang, X. Cheng, H. Wang, L. Yang, Mobile demand forecasting via deep graph-sequence spatio-temporal modeling in cellular networks, IEEE Internet of Things Journal 5 (4) (2018) 3091–3101.

[37] Y. Lu, Y. Shu, X. Tan, Y. Liu, M. Zhou, Q. Chen, D. Pei, Collaborative learning between cloud and end devices: an empirical study on location prediction, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 139–151.

[38] X. Zhang, M. Qiao, L. Liu, Y. Xu, W. Shi, Collaborative cloud-edge computation for personalized driving behavior modeling, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 209–221.

[39] H. Daga, P. K. Nicholson, A. Gavrilovska, D. Lugones, Cartel: A system for collaborative transfer learning at the edge, in: Proceedings of the ACM Symposium on Cloud Computing, 2019, pp. 25–37.