ELASTIC: Edge Workload Forecasting based on Collaborative Cloud-Edge Deep Learning

Yanan Li
Beijing University of Posts and Telecommunications
Beijing, China
YaNanLi@bupt.edu.cn

Haitao Yuan*
Nanyang Technological University
Singapore, Singapore
yht19@tsinghua.org.cn

Zhe Fu
Tsinghua University
Beijing, China
zhefu0@outlook.com

Xiao Ma*
Beijing University of Posts and Telecommunications
Beijing, China
maxiao18@bupt.edu.cn

Mengwei Xu
Beijing University of Posts and Telecommunications
Beijing, China
mwx@bupt.edu.cn

Shangguang Wang
Beijing University of Posts and Telecommunications
Beijing, China
sgwang@bupt.edu.cn

ABSTRACT

With the rapid development of edge computing in the post-COVID19 pandemic period, precise workload forecasting is considered the basis for making full use of the edge limited resources, and both edge service providers (ESPs) and edge service consumers (ESCs) can benefit significantly from it. Existing paradigms of workload forecasting (i.e., edge-only or cloud-only) are improper, due to failing to consider the inter-site correlations and might suffer from significant data transmission delays. With the increasing adoption of edge platforms by web services, it is critical to balance both accuracy and efficiency in workload forecasting. In this paper, we propose ELASTIC, which is the first study that leverages a cloud-edge collaborative paradigm for edge workload forecasting with multi-view graphs. Specifically, at the global stage, we design a learnable aggregation layer on each edge site to reduce the time consumption while capturing the *inter-site* correlation. Additionally, at the local stage, we design a disaggregation layer combining both the *intra-site* correlation and *inter-site* correlation to improve the prediction accuracy. Extensive experiments on realistic edge workload datasets collected from China's largest edge service provider show that ELASTIC outperforms state-of-the-art methods, decreases time consumption, and reduces communication cost.

CCS CONCEPTS

• **Computer systems organization → Distributed architectures**.

KEYWORDS

Workload forecasting, cloud-edge collaboration, edge computing

ACM Reference Format:
Yanan Li, Haitao Yuan, Zhe Fu, Xiao Ma, Mengwei Xu, and Shangguang Wang. 2023. ELASTIC: Edge Workload Forecasting based on Collaborative Cloud-Edge Deep Learning. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583436

1 INTRODUCTION

In the post-COVID19 pandemic period, more and more web services are adopting edge computing due to the advantages of low latency and high bandwidth, e.g., web AR/VR [27], web gaming [22] and many web of things (WoT) applications [1]. Although edge computing is gaining momentum, *making the most of limited edge resources has always been a major problem* [39]. One basic way to solve this problem is to accurately predict future workloads at each edge site [25], which benefits both edge service providers (ESPs) and edge service consumers (ESCs). Specifically, (i) For ESPs, the proactive estimation of future workload has become a solution to many critical challenges such as resource management [3, 26]. For example, accurate workload forecasting and scheduling algorithms are expected to reduce ESP's monetary bandwidth costs by up to 65% [31]. (ii) For ESCs, by the accurate workload forecasting, they have opportunities to promote the quality of service (QoS) for end-users [13]. For example, the response time of web services achieves more than 30% promotion with an outlier-resilient QoS forecasting method [40]. Notably, we explicitly target the resource usage workloads (CPU utilization, outbound bandwidth, etc.) of virtual machines (VMs) as it is the mainstream form of web subscription to edge platforms [9].

There are two typical paradigms of workload forecasting: edge-only [10] and cloud-only [17, 46]. With the increasing adoption of edge platforms by web services, it is difficult for the above paradigms to balance both accuracy and efficiency. Edge-only refers to deploying the forecasting model at each edge site to capture the intra-site correlation among applications. Although this approach looks plausible and easily executable, it **does not (or poorly) consider the inter-site correlations**, which might provide useful information for more accurate workload forecasting. For example, VMs deployed by the same ESC at different edge sites may have similar diurnal workload patterns [39], so we can leverage the pattern

[*] Corresponding authors: Haitao Yuan and Xiao Ma.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583436

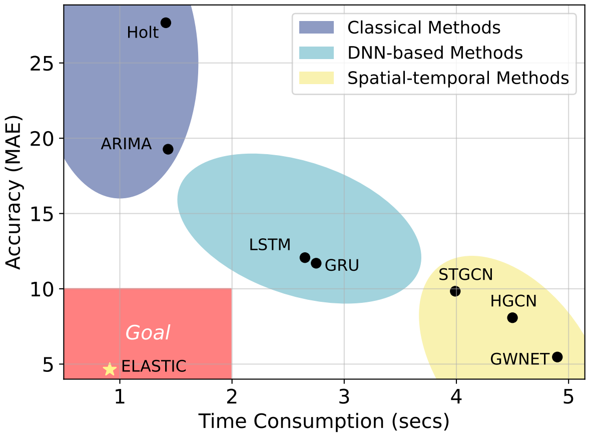

Figure 1: The accuracy-efficiency tradeoff of three types of forecasting methods on the Cloud-only paradigm. Accuracy is determined by the average MAE of 12 prediction horizons validated on our bandwidth dataset (§4). Efficiency is determined by the time consumption, which includes two parts: the data transmission delay and the model inference time. The networking environment of data transmission delay is uniformly set to medium (16 Mbps).

similarity to boost prediction accuracy. Cloud-only refers to all edge sites transmitting the whole workload data to the centralized cloud to facilitate centralized model training and inference. This centralized, cloud-only approach has some inherent limitations. First, data transmission and the large centralized model might **add additional time overhead to both the training and inference phase** [10], which is more significant as the number of VMs increases and the networking environment deteriorates. As depicted in Figure 1, existing methods hardly satisfy both accuracy and efficiency in the cloud-only paradigm. Second, it **imposes excessive traffic on the backhaul network**, which is already overburdened [39].

To address the aforementioned challenges, in this paper, we propose a novel Edge workLoad forecASting framework based on a collaboraTIve Cloud-edge paradigm, namely ELASTIC. It mainly consists of two stages: (1) the global stage, which performs coarse-grained spatial-temporal forecasting at a centralized cloud for capturing the *inter-site* correlations (i.e., at edge site granularity) with the data aggregated by each edge site; (2) the local stage, which performs fine-grained spatial-temporal forecasting at each edge site for capturing the *intra-site* correlations (i.e., at VM/web service granularity) between different workloads. The final forecasting results combine both the intra-site correlations and inter-site correlations.

A well-defined graph is essential to the success of the spatial-temporal model, but it is indirect, especially at different stages. The indirect graph implies that, unlike the distance graph in traffic flow prediction, we can hardly capture actual dependence correlations at different stages [21]. Specifically, the indirection of graphs includes two aspects: complicated and implicit correlations. To address this issue, we construct multi-view graphs at the global/local stage to incorporate complicated and implicit correlations between edge sites/VMs, which might provide significant information for workload forecasting. Specifically, at the global stage, we consider spatial, temporal, and self-adaptive graphs. The spatial graph represents various distances with respect to the spatial dimension, such as geographic distance (km) and network distance (RTT). The temporal graph depicts the DTW (dynamic time warping [28])-measured workload similarity. The self-adaptive graph, represented by a learned adjacency matrix [38], reflects edge sites' implicit correlation. At the local stage, we explore physical, logical, and self-adaptive graphs. The physical and logical graphs indicate similarities in hardware (e.g., CPU cores, RAM) and software (i.e., whether deployed by the same users). The self-adaptive graph reflects the implicit correlation among applications at each edge site.

Finally, in contrast to the previous edge workload forecasting with synthetic datasets [23, 25], the evaluation of ELASTIC is based on real-world edge workload datasets collected by NEP (Next-generation Edge Platform) [39], one of China's largest edge service providers. We select partial datasets that contain the CPU and bandwidth workloads of 1,281 IaaS VMs (applications) at 43 edge sites from June 2020 to August 2020. Moreover, when compared with mainstream time series forecasting methods, the results show that ELASTIC outperforms them on both datasets, reduces by 91.24% communication cost on average, and decreases both the training time and inference time significantly. Ablation experiments and comparative experiments have further confirmed the rationality and effectiveness of our framework.

In summary, the major contributions of this paper are as follows:

- To the best of our knowledge, this is the first study that leverages the collaborative cloud-edge paradigm for edge workload forecasting based on spatial-temporal models.
- We propose ELASTIC, a novel two-stage framework with multi-view graphs for edge workload forecasting. It not only captures the intra-site spatial-temporal correlations among workloads but also the inter-site spatial-temporal correlations among edge sites in a decentralized manner.
- Extensive experimentation is conducted utilizing the real-world edge workload traces collected by one of China's largest edge service providers-NEP. The evaluation results demonstrate that ELASTIC significantly outperforms the state-of-the-art methods, decreases time consumption, and reduces communication cost.

2 PRELIMINARIES

To begin with, we define some important notations, and then mathematically restate the edge workload forecasting problem.

Edge workloads. The public edge platform under consideration consists of $\mathcal{M} = \{1, ..., m, ..., M\}$ edge sites. There are a total of $\mathcal{N} = \{1, ..., n, ..., N\}$ VMs among M edge sites and N_m VMs on top of each edge site. For each VM n at the time interval t, it has three types of workloads c_n^t, d_n^t, u_n^t, which represent CPU utilization, upstream bandwidth, and downstream bandwidth of VM n in time interval t, respectively. Since user requests (downstream bandwidth of VM) tend to be much smaller than server replies (upstream bandwidth of VM) for online services, the outgoing upstream bandwidth often dominates [44]. Therefore, we mostly consider CPU and upstream bandwidth workloads here. For sake of generality, we use $\mathbf{x}_n^t$ to refer to any kind of workload for VM n at time interval t.

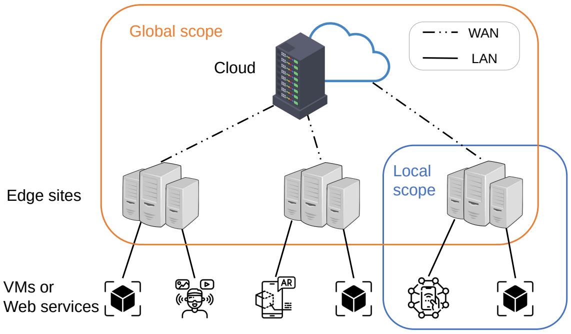

Figure 2: Cloud-Edge Architecture.

Global scope. From a higher viewpoint, we can construct a graph $\mathcal{G}^{global} = (\mathcal{M}, \mathcal{E}^{global})$ to represent the spatial dependencies among M edge sites. $\mathcal{E}^{global}$ is the set of graph edges where each element represents the correlation between each pair of edge sites. We describe in detail how to construct different graphs for capturing different edge site correlations in Section 3.

Local scope. Within each edge site, we can also construct a graph $\mathcal{G}_m^{local} = (\mathcal{V}_m, \mathcal{E}_m)$ to represent the spatial dependencies among N_m VMs, where $\mathcal{V}_m (|\mathcal{V}_m| = N_m)$ is the set of VMs belonging to edge site m. Similarly, $\mathcal{E}_m$ is the set of graph edges where each element represents the correlation between each pair of VMs. According to different guidelines, we can also construct different graphs with different $\mathcal{E}_m$ (§3). In addition, as shown in Figure 2, the connection between VMs (applications) in the local scope is through the local area network (LAN), which is different from the wide area network (WAN) between edge sites in global scope.

Edge workload forecasting. Given the local scope graphs, global scope graphs and previous P timestamps workloads' records $X^{t-P}, \ldots, X^{t-1} (X^t = \{\mathbf{x}_1^t, \ldots, \mathbf{x}_N^t\} \in \mathbb{R}^N)$, the edge workload forecasting aims to build a model F with parameter Φ, which predicts the next H time steps values of the corresponding workload, denoted by $\widetilde{X}^t, \widetilde{X}^{t+1}, \ldots, \widetilde{X}^{t+H-1}$.

$$\widetilde{X}^t, \widetilde{X}^{t+1}, \ldots, \widetilde{X}^{t+H-1} = F(X^{t-P}, \ldots, X^{t-1}; \mathcal{G}; \Phi) \tag{1}$$

3 METHODOLOGY

In this section, we first introduce the overall architecture of our system, and then describe the specific design of the local and global stage, respectively.

3.1 Framework Overview

As shown in Figure 3, ELASTIC mainly consists of two stages: the global stage and local stage. The global stage, which performs coarse-grained spatial-temporal forecasting with global multi-view graph at centralized cloud for capturing the inter-level correlations (i.e., edge site granularity) with the data aggregated by each edge site. Similarly, the local stage, which performs fine-grained spatial-temporal forecasting with local multi-view graph at each edge site for capturing the intra-level correlations (i.e., VM/web service granularity) among different applications within each edge site. Through the Disaggregation layer, we can get the final forecasting results, which combine both the intra-level correlations and inter-level correlations. In this way, ELASTIC not only improves the prediction accuracy, but also reduces the bandwidth consumption compared to centralized cloud-only paradigm.

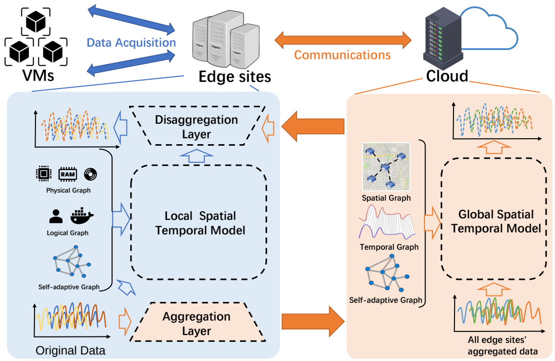

Figure 3: Overall architecture of our system.

3.2 Global Stage

The global stage consists of aggregation layers distributed on each edge site and the global spatial-temporal prediction model on the centralized cloud. The input of the global spatial-temporal prediction model consists of the output of all edge sites' aggregation layer and the global multi-view graphs, which represent the correlation among edge sites.

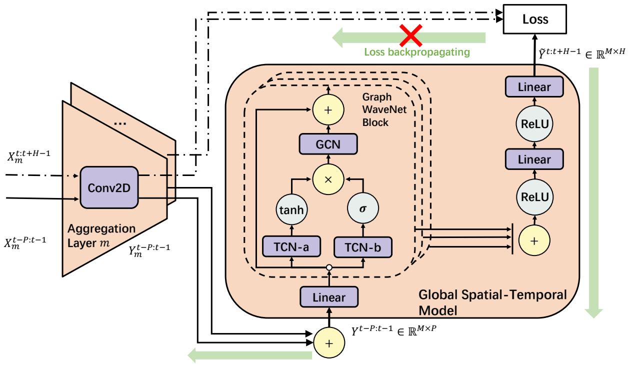

Figure 4: Architecture of the global spatial-temporal prediction model.

Aggregation layer on each edge site. As shown in Figure 4, we deploy an aggregation layer on each edge site to compress raw data for the reduction of bandwidth consumption of global spatial-temporal model (GSTM) training on the centralized cloud. Each edge site's aggregation layer and the GSTM are trained jointly. Specifically, the input of aggregation layer on each edge site is $X_m^{t-P:t-1} \triangleq \{X^{t-P}, \ldots, X^{t-1}\} \in \mathbb{R}^{N_m \times P}$ and the output is $Y_m^{t-P:t-1} = W_m X_m^{t-P:t-1} \in \mathbb{R}^{1 \times P}$. The aggregated results $Y_m^{t-P:t-1}$ from all edge sites are concatenated together and then supplied into the global spatial-temporal prediction model as input. For the global spatial-temporal prediction model, not only the input, but also the prediction target used to compute the loss passes through the same aggregation layer of each edge site. When backpropagating, the aggregation layer only obtains gradients from the input of GSTM, not from the aggregated prediction target.

Global multi-view graphs. The complicated correlation among edge sites can also be divided into three views: spatial, temporal, and self-adaptive view. The spatial view graph $\mathbf{A}^{spatial}$ is used to reflect the physical proximity between edge sites (e.g., distances or RTT), while the temporal view graph is used to reflect the similarity of aggregated workload data between edge sites. Here we use dynamic time warping (DTW) [28] to measure the temporal similarity $A_{i,j}^{temporal}$ between edge site i and edge site (location) j for constructing the temporal view graph. To control the sparsity of spatial view graph and temporal view graph, we apply following function to both of them.

$$\mathcal{A}_{i,j} = \begin{cases} \exp\left(-\frac{d_{i,j}^2}{\sigma^2}\right), \text{ if } d_{i,j} \leq \kappa \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

where $d_{i,j}$ is the distance from edge site i to j (e.g., $RTT(i,j)$ for spatial view graph and $DTW(i,j)$ for temporal view graph), σ is the standard deviation, and κ is the threshold to control the sparsity.

In order to discover the implicit correlation among edge sites, we adopt a global self-adaptive adjacency matrix [38]. This global self-adaptive adjacency matrix is learned end-to-end using stochastic gradient descent and does not require any prior knowledge. This allows the model to uncover hidden correlation among edge sites on its own. To do this, we randomly initialize two site embedding vectors with learnable parameters $E_1, E_2 \in R^{M \times c}$ and the adjacency matrix of this global self-adaptive view can be calculated by:

$$\mathbf{A}^{global\_adaptive} = \text{SoftMax}\left(\text{ReLU}\left(\mathbf{E}_1 \mathbf{E}_2^T\right)\right) \tag{3}$$

where we apply the ReLU activation function to eliminate weak connections, and the SoftMax function to normalize the global self-adaptive adjacency matrix.

Global spatial-temporal model. The global spatial-temporal model is mainly composed of K blocks. Each block consists of a gated temporal convolution layer (TCN) module and a graph convolutional network (GCN) layer. The gated TCN module [42], which is made up of two parallel TCN layers and their results are passed through a gating mechanism to produce the final result, takes the form of:

$$\mathbf{h} = g\left(\mathbf{\Theta}_1 \star \mathcal{X} + \mathbf{b}\right) \odot \sigma\left(\mathbf{\Theta}_2 \star \mathcal{X} + \mathbf{c}\right) \tag{4}$$

where $\mathbf{\Theta}_1$, $\mathbf{\Theta}_1$, $\mathbf{b}$ and $\mathbf{c}$ are model parameters, $\odot$ is the element-wise product, $g(\cdot)$ is an tangent hyperbolic activation function of the outputs, and $\sigma(\cdot)$ is the sigmoid function which determines the ratio of information passed to the next layer.

The graph convolutional network (GCN) layer is an useful operation to extract spatial correlations given its structural information. We adopt a truncated expansion of GCN in terms of Chebyshev polynomials to the first order [15].

$$\mathbf{h}' = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \mathbf{h} \Theta \tag{5}$$

where D is the graph degree matrix, and A represents the graph adjacency matrix. Furthermore, we normalize them through the equation $\tilde{A} = A + I_N, \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and I_N is the identity matrix.

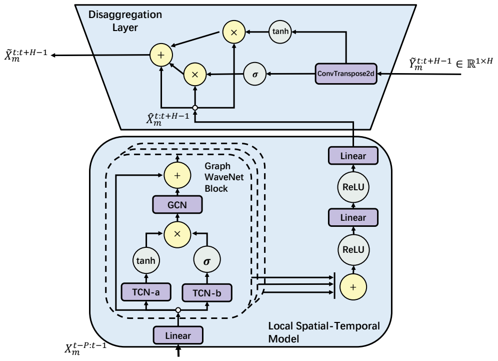

Figure 5: Architecture of the local spatial-temporal prediction model.

3.3 Local Stage

As is shown in Figure 5, the local stage of each edge site consists of the local spatial-temporal prediction model and disaggregation layer. As mentioned earlier, the local spatial-temporal prediction model is responsible for learning the correlations of workload records between different VMs on the same edge site (spatial) and the autocorrelation for each VM at different time slots (temporal). Similar to the global stage, the graphs input to the local spatial-temporal prediction model are also multi-view graphs.

Local multi-view graphs. Specifically, the complicated local multi-view graphs consists of three parts: physical, logical, and local self-adaptive graph. Physical graph is constructed based on the similarity between different VMs in terms of CPU cores, memory size, and storage size. For each VM, we use the CPU cores, memory size, and storage size to form a vector. The physical similarity between two VMs is defined as the cosine similarity of the two vectors. Similarly, logical graph represents the similarity between pairs of VMs in terms of logical features, such as the userID and the imageID. We first encode the userID and imageID using the *One-Hot Encoding* and then use the cosine similarity to measure the similarity between each pair VMs on encoded vectors. The local self-adaptive graph is similar to the global self-adaptive graph, so it is omitted here without much elaboration.

Disaggregation Layer. The disaggregation layer is responsible for combining the output of the local spatial-temporal prediction model $\hat{X}_m^{t:t+H-1}$ with the inference result of the global spatial-temporal prediction model $\widetilde{Y}_m^{t:t+H-1}$ and generates the final prediction result $\widetilde{X}_m^{t:t+H-1}$. Specifically, $\widetilde{Y}_m^{t:t+H-1}$ passes through a linear layer implemented by ConvTranspose2D to produce $\overline{X}_m^{t:t+H-1}$, which is then merged with $\hat{X}_m^{t:t+H-1}$ through gating mechanism to produce the final prediction $\widetilde{X}_m^{t:t+H-1}$.

$$\begin{aligned} \widetilde{X}_m^{t:t+H-1} = \omega_0 * \hat{X}_m^{t:t+H-1} + \omega_1 * g(\overline{X}_m^{t:t+H-1}) \odot \hat{X}_m^{t:t+H-1} + \\ \omega_2 * \sigma(\overline{X}_m^{t:t+H-1}) \odot \hat{X}_m^{t:t+H-1} \end{aligned} \tag{6}$$

$$\overline{X}_m^{t:t+H-1} = \Theta \star \widetilde{Y}_m^{t:t+H-1} + b \tag{7}$$

where ω_0, ω_1 and ω_2 are learnable parameters initialized to 1, 0, 0. $\odot$ is the element-wise product. The activation function $g(\cdot)$ is empirically set to the tangent hyperbolic function, and $\sigma(\cdot)$ is the sigmoid function that controls the ratio of information passed to the next layer. This approach is capable of capturing both the linear and nonlinear relationship between $\hat{X}_m^{t:t+H-1}$ and $\widetilde{Y}_m^{t:t+H-1}$. Θ and b are parameters of ConvTranspose2D, where Θ is initialized by the corresponding parameters of Conv2d in the Aggregation Layer trained at the global stage of each edge site.

Local spatial-temporal model. The specific design of the local spatial-temporal model is similar to the global spatial-temporal model. The only difference is that considering the relatively limited resources of edge sites, the number of blocks that make up the local spatial-temporal model is smaller.

3.4 Complexity Analysis

The specific training pipeline of ELASTIC's global stage and local stage is outlined in Appendix A. Moreover, we also compare the ELASTIC framework with traditional cloud-based centralized spatial-temporal prediction models in terms of time complexity. Since the current mainstream spatial-temporal prediction methods mainly rely on graph neural networks, according to [37], the time complexity of these methods is $O(N^2)$ in the worst case. Therefore, for ELASTIC, its time complexity is $M * O(\overline{N_m}^2) + O(M^2)$, where M is the number of edge sites, $\overline{N_m}$ is the average number of VMs of each edge site, and N is the total number of VMs among M edge sites. Since $M \ll N$, the time complexity of ELASTIC $O(N\overline{N_m})$ is similar to edge-only spatial-temporal model, which is smaller than the centralized-cloud spatial-temporal model $O(N^2)$.

4 EXPERIMENTAL SETTINGS

4.1 The Dataset

Our model evaluation is based on a large-scale real-world dataset collected from NEP (**N**ext-generation **E**dge **P**latform) [39], one of China's largest edge service providers. Compared with traditional cloud computing, NEP has two main differences. First, NEP leverages the resources of ISPs to build miniaturized datacenters (edge sites) with computation, network, and storage capabilities. So as opposed to cloud data centers, each edge site only has one outbound ISP. Secondly, traditional cloud platforms typically have less than ten data centers in one country. In contrast, NEP's edge sites number is about two orders of magnitude larger (130+ in China), and the number is still fast-growing. In spite of the increasing number of edge sites, the capacity of each edge site is relatively limited compared to traditional cloud data centers. While NEP supports many types of services (e.g., PaaS and FaaS), the current dominant usage is Infrastructure-as-a-Service (IaaS) VMs. Thus, this paper mainly targets IaaS VMs hosted in NEP for workload forecasting.

The following experiments are mainly conducted on two datasets that contain 43 edge sites and 1,281 VMs running on NEP from June 2020 to August 2020. Specially, it includes: (1) the average CPU utilization per 5 minutes of each VM; (2) the average upstream and downstream bandwidth (Mbps) per 5 minutes of each VM. We adopt Z-score normalization to process the data in both datasets. Moreover, we split each dataset into the training, validation, and test sets in the proportion of 7:2:1. We train models in the training-set, and according to the results of the validation-set choose the optimal parameters to test the model on the test-set. It is noted that missing values are excluded in both cases from training-set, validation-set and test-set.

4.2 Evaluation Baselines

To evaluate the performance of ELASTIC, we employ several common time series forecasting methods from three categories for comparison: (1) the conventional time series forecasting methods (i.e., ARIMA [2]); (2) recurrent neural network based methods (i.e., LSTM [30]); (3) recent advanced spatial-temporal forecasting methods (i.e., GraphWaveNet [38], DCRNN[20], ASTGCN [8], GCRN [29], STGCN [41], HGCN[7], OGCRNN [5], OTSGGCN [6]). See Appendix B for more details. Unless otherwise specified, these baselines are implemented using the cloud-only paradigm since it is generally more accurate than the edge-only paradigm.

4.3 Evaluation Metrics

To evaluate the prediction performance, we employ three evaluation metrics. The following is a brief description of these metrics:

- The mean absolute error (MAE) reflects the average of the absolute errors.
$$MAE = \frac{1}{H * N} \sum_{j=1}^{H} \sum_{i=1}^{N} \left| \hat{y}_{ij} - y_{ij} \right| \tag{8}$$
- The mean squared error (MSE) reflects the squared difference between the predicted and the ground truth values.
$$MSE = \frac{1}{H * N} \sum_{j=1}^{H} \sum_{i=1}^{N} \left| \hat{y}_{ij} - y_{ij} \right|^2 \tag{9}$$
- The symmetric mean absolute percentage error (SMAPE) reflects the percentage of the error to the ground-truth value. Since it is scale-independent, prediction errors are considered regardless of the magnitude of the sequence, which is particularly important for bandwidth dataset.
$$SMAPE = \frac{1}{H * N} \sum_{j=1}^{H} \sum_{i=1}^{N} \frac{|\hat{y}_{ij} - y_{ij}|}{(|y_{ij}| + |\hat{y}_{ij}|)/2} * 100\% \tag{10}$$

where N is the number of VMs, H is the prediction horizon and $\hat{y}_{ij}$ is the forecast value of the ground truth y_{ij}. For all three of them, the lower value represents the higher accuracy of prediction.

4.4 Implementation Details

All the models are trained and evaluated upon a Linux server with Intel(R) Xeon(R) Gold 5218R @ 2.10GHz, 819GB RAM, and NVIDIA Tesla-V100. To simulate slow, medium, and fast network environments, similar to [43], we throttle the uplink speed between edge sites and cloud to 4 Mbps, 16 Mbps, and 50 Mbps respectively.

The specific settings of global multi-view graphs' threshold are outlined in Appendix C.1. The input sequence length P is 12, and the prediction horizon H is set to 1, 4, 8, and 12, respectively. The different prediction horizons indicate predicting at different granularities. Our proposed methods are optimized with Adam optimizer [14], and its learning rate starts from 0.001, decaying with the *ExponentialLR* learning rate scheduler. To train all methods, we choose the mean square error (MSE) as the loss function. The batch size is 64 and the maximum training epoch for both the global stage and local stage is 50. We run each approach five times and report the average results.

Table 1: Performance comparison of ELASTIC and other baselines on CPU dataset. The numbers on the first line (1/4/8/12) represent different time periods to forecast. MAE/SMAPE/MSE represents different evaluation metrics, respectively.

Models	1			4			8			12		
	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE
ARIMA	0.361	46.86%	1.133	0.567	65.52%	3.144	0.583	67.65%	3.320	0.596	69.70%	3.322
LSTM	0.180	30.47%	0.408	0.257	33.58%	0.764	0.339	34.30%	1.175	0.406	34.24%	1.533
GraphWaveNet	0.114	21.65%	0.262	0.152	23.80%	0.327	0.189	26.32%	0.498	0.215	28.10%	0.652
DCRNN	0.121	26.24%	0.208	0.158	26.98%	0.373	0.220	29.27%	0.647	0.277	30.57%	0.941
ASTGCN	0.183	28.43%	0.281	0.196	28.85%	0.376	0.222	31.76%	0.546	0.251	32.39%	0.754
GCRN	0.178	30.56%	0.321	0.219	32.39%	0.521	0.273	35.68%	0.838	0.325	39.06%	1.185
STGCN	0.160	31.65%	0.278	0.206	34.45%	0.490	0.266	37.94%	0.822	0.323	40.80%	1.194
HGCN	0.130	27.18%	0.186	0.167	31.04%	0.335	0.206	35.46%	0.518	0.236	37.76%	0.693
OGCRNN	0.215	28.02%	0.428	0.247	30.07%	0.592	0.287	36.37%	0.841	0.325	39.01%	1.102
OTSGGCN	0.115	20.57%	0.190	0.168	23.99%	0.406	0.231	27.93%	0.716	0.285	31.65%	1.044
ELASTIC (ours)	**0.112**	**20.10%**	**0.183**	**0.143**	**20.94%**	**0.319**	**0.174**	**22.93%**	**0.477**	**0.197**	**24.06%**	**0.622**

Table 2: Performance comparison of ELASTIC and other baselines on bandwidth dataset. Note that the unit of MAE is Mbps.

Models	1			4			8			12		
	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE	MAE	SMAPE	MSE
ARIMA	13.12	55.39%	1727.54	17.67	59.95%	3400.80	20.09	60.22%	4510.45	20.18	61.84%	4603.40
LSTM	11.96	49.46%	3323.55	12.15	54.28%	3551.53	12.69	58.93%	3886.18	13.67	60.82%	4186.95
GraphWaveNet	2.86	41.45%	176.02	4.24	44.75%	439.08	5.85	45.16%	804.27	7.54	52.18%	1179.19
DCRNN	5.68	47.73%	723.19	6.49	50.68%	940.03	7.46	53.37%	1073.55	7.90	56.91%	1493.86
ASTGCN	10.34	46.18%	1244.95	11.29	51.54%	1293.49	12.54	51.82%	1458.01	13.52	58.68%	1771.76
GCRN	9.97	54.11%	2951.97	10.77	56.57%	3157.50	12.02	58.29%	3487.31	13.22	60.21%	3809.47
STGCN	3.63	53.28%	223.28	5.16	54.78%	558.23	7.11	55.04%	1048.87	8.90	57.36%	1579.65
HGCN	3.21	41.84%	204.46	4.48	44.28%	464.11	5.91	47.14%	758.45	7.11	49.78%	1005.77
OGCRNN	10.88	44.45%	2989.86	11.48	51.96%	3194.18	12.31	55.27%	3505.67	13.08	50.28%	3785.13
OTSGGCN	3.01	47.38%	177.28	4.33	49.35%	502.75	6.15	52.73%	909.69	7.75	55.77%	1300.30
ELASTIC (ours)	**2.72**	**26.62%**	**175.24**	**3.91**	**38.06%**	**413.64**	**5.16**	**44.98%**	**718.44**	**6.17**	**48.92%**	**960.98**

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we evaluate the proposed ELASTIC framework on real-world datasets for workload forecasting, and present the experimental results as compared with different categories of competitive techniques. Particularly, we aim to answer the following research questions via the experiments:

- **RQ1**: Compared with state-of-the-art forecasting models, how does ELASTIC perform in workload forecasting with respect to different prediction horizons?
- **RQ2**: How do different modules and specific settings affect the prediction performance of ELASTIC?
- **RQ3**: When our framework is combined with other forecasting methods, is there still a performance improvement?
- **RQ4**: What is the actual time consumption and communication cost of implementing ELASTIC?

5.1 Overall Performance (RQ1)

Table 1 and Table 2 show the forecasting accuracy of two kinds of workloads (CPU and bandwidth) with respect to different time periods in terms of MAE, SMAPE, and MSE. From the evaluation results, we summarize two key observations as follows:

First and foremost, ELASTIC is significantly better than other different types of neural network-based methods. For example, on bandwidth datasets, ELASTIC achieves relatively 0.14, 14.83%, and 0.78% improvements over the best-performed baseline (i.e., Graph WaveNet) in terms of MAE, SMAPE, and MSE on one prediction horizon. This sheds light on the benefit of our two-stage multi-view model which jointly considers the inter-site correlations and intra-site correlations.

Second, neural network-based forecasting methods are superior to the conventional time series forecasting techniques (i.e., ARIMA). This is due to the facts that: (1) Conventional time series forecasting methods emphasize a single fixed temporal pattern rather than time-evolving temporal relationships; (2) It is more advantageous to utilize neural network-based methods in order to capture the inherent correlations of multidimensional spatial-temporal data in a nonlinear way.

5.2 Ablation Studies of ELASTIC (RQ2)

In addition to comparing ELASTIC with state-of-the-art methods, we also aim to get a better understanding of key components of ELASTIC by studying the effectiveness of different components. The ablation studies are also divided into local and global stages.

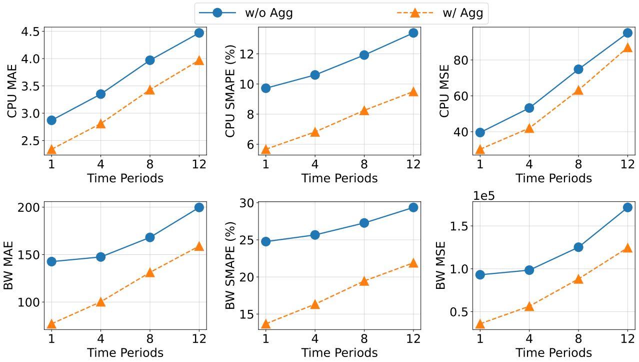

Figure 6: Performance comparison of ELASTIC's global stage with (w/ Agg) and without aggregation layer (w/o Agg).

Global Stage. At the global stage, our ablation studies are mainly focused on two parts, the aggregation layer on each edge site and the global multi-view graphs. Figure 6 demonstrates the evaluation results of ELASTIC's global stage with and without aggregation layer on each edge site. We can notice that the full version of ELASTIC achieves the best performance in all cases. For this reason, adding an aggregation layer to each edge site is necessary in order to effectively aggregate the data rather than simply adding it, improving the accuracy of the global spatial-temporal model.

Additionally, we also conduct experiments with ELASTIC using five different adjacency matrix configurations to verify the effectiveness of our global multi-view graphs. Table 3 shows the average score of MAE, SMAPE, and MSE over 12 prediction horizons. We find that the adaptive-only model works best among identity, spatial-only, and temporal-only on all three evaluation metrics. It indicates that even in the absence of an explict graph structure, self-adaptive adjacency matrices can produce good prediction performance. The spatial-temporal-adaptive model achieves the lowest scores on all three evaluation metrics. It implies that if graph structure information is provided, adding the self-adaptive adjacency matrix to the model could bring new and relevant information.

Table 3: Performance comparison of ELASTIC's global stage with different adjacency matrix configurations.

		Mean MAE	Mean SMAPE	Mean MSE
CPU	Identity	4.02	9.99%	90.24
	spatial-only	4.01	9.82%	89.50
	temporal-only	3.90	9.62%	86.96
	adaptive-only	3.32	9.14%	67.14
	spatial-temporal -adaptive	**3.14**	**7.56%**	**55.63**
BW	Identity	181.09	31.70%	139927
	spatial-only	147.06	27.28%	117785
	temporal-only	159.69	33.91%	126999
	adaptive-only	131.19	26.04%	81529
	spatial-temporal -adaptive	**116.87**	**17.85%**	**76206**

Local Stage. Similarly, the ablation studies of ELASTIC's local stage are also focused on two parts, the disaggregation layer and local multi-view graphs. Figure 7 demonstrates the evaluation results of ELASTIC's local stage with and without disaggregation layer on each edge site. We can notice that the full version of ELASTIC's local stage achieves the best performance in all cases. For this reason, adding an disaggregation layer to each edge site is necessary to fine tune the prediction results of local spatial-temporal model, improving the model accuracy. Due to space limitations, the ablation studies of local multi-view graphs are given in Appendix C.2.

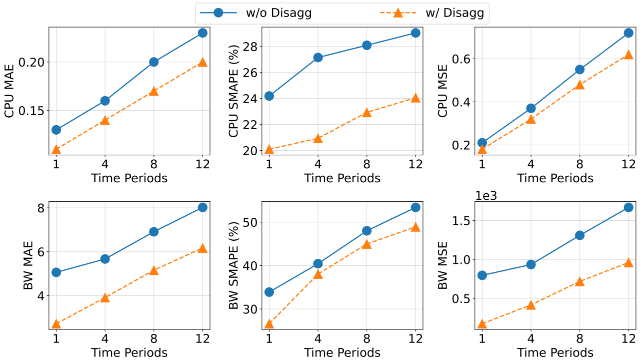

Figure 7: Performance comparison of ELASTIC's local stage with (w/ Disagg) and without disaggregation layer (w/o Disagg) on CPU and bandwidth datasets.

5.3 Specific Settings (RQ2) and Further Practice (RQ3)

Table 4 demonstrates the evaluation results of ELASTIC's local stage with and without initializing the ConvTranspose2D by the corresponding parameters of Conv2d in the global stage's Aggregation Layer on the average score of MAE, SMAPE, and MSE over 12 prediction horizons. We can observe that the full version of ELASTIC achieves the best performance in all evaluation metrics. It demonstrates that the use of applying the Aggregation Layer parameters learned at the global stage to the Disaggregation Layer in the local stage is effective. Moreover, we further apply our framework with different methods in Appendix C.3.

5.4 Time Consumption and Communication Cost (RQ4)

Figure 8 and Figure 9 compare the ELASTIC with centralized Graph WaveNet at cloud over training time and inference delay in slow (4 Mbps), medium (16 Mbps), and fast (50 Mbps) networking environments. When the network environment between edge sites and centralized cloud is worse, the difference in time consumption between ELASTIC and centralized Graph WaveNet is more significant. For example, when the network condition is slow, the training time of ELASTIC and centralized Graph WaveNet on the CPU dataset is 5.46/15.67 (8.50/24.17 on the bandwidth dataset), and the inference time is 0.94/4.59 (1.34/6.43 on the bandwidth dataset), respectively. In general, ELASTIC has a shorter training time and inference delay than Graph WaveNet since it needs to transmit fewer data between edge sites and the cloud center through the network. Specifically, the communication cost of ELASTIC and centralized Graph WaveNet on the CPU dataset is 116.89MB/1337.109MB

(180.18MB/2053.418MB on the bandwidth dataset), respectively. The communication cost of ELASTIC is decreased by 91.24% on average when compared to the centralized Graph WaveNet.

Table 4: Performance comparison of ELASTIC with (w/ Initial) and without initializing (w/o Initial) the ConvTranspose2D by the corresponding parameters of Conv2d at the global stage's Aggregation Layer on the average of MAE, SMAPE, and MSE over 12 prediction horizons.

		Mean MAE	Mean SMAPE	Mean MSE
CPU	w/o Initial	0.164	22.90%	0.413
	w/ Initial	**0.152**	**21.65%**	**0.400**
BW	w/o Initial	4.94	47.62%	679.19
	w/ Initial	**4.52**	**37.03%**	**588.27**

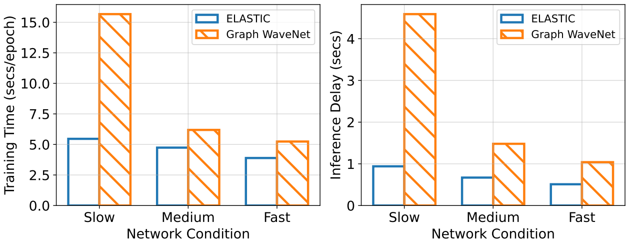

Figure 8: The comparison of training time and inference delay for ELASTIC and Graph WaveNet in slow, medium, and fast networking environments on CPU dataset.

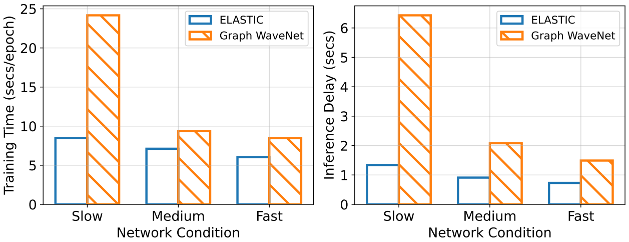

Figure 9: The comparison of training time and inference delay for ELASTIC and Graph WaveNet in slow, medium, and fast networking environments on bandwidth dataset.

6 RELATED WORK

Time Series Prediction. Time series prediction has been a longstanding problem [21]. The classic methods are primarily concerned with forecasting for each individual time series. For example, Holtwinter [36] and ARIMA [2] study the linear relationship between past observations and the future of each time series. However, since they adopt a linear approach to predicting the future, the prediction accuracy is generally poor. With the recent advancement of deep learning, neural network-based models have gained considerable attention for non-linear prediction. Among them, the most widely used are RNN-based models, such as LSTM [30] and GRU [33]. Since they process data in a recurrent fashion, they generally need to spend more time and hardware resources on training. Moreover, the Attention-based methods are also very popular recently [32, 45], which mainly utilize multi-head attention mechanism for capturing the potential relationship. However, the encoder-decoder architecture of Attention-based methods results in huge model storage overhead, is not suitable for resource-constrained edge sites and do not (or poorly) consider the spatial correlations between different applications [35].

Spatial-Temporal Prediction. Spatial-temporal predictions that consider the impact of time series themselves (temporal) and the mutual impact between time series (spatial) have received substantial attention in recent years [7, 20, 38, 41]. However, they mainly focus on traffic flow prediction using the PEMS-BAY and METR-LA datasets [20], and rarely on workload forecasting, especially in edge scenarios. Although some related work considers workload forecasting, they mainly utilize synthetic datasets [23, 25] rather than real edge workload datasets and target to centralized-cloud scenarios [17, 18, 46]. Furthermore, the mainstream spatial-temporal prediction methods are trained in a centralized manner. As the number of VMs on the edge sites increases, the direct implementation of centralized training for workload forecasting in edge scenario may lead to huge model storage overhead, significant bandwidth consumption, and non-negligible transmission delay.

Cloud-Edge Collaboration. With the development of edge computing and overcoming the limitations of centralized cloud computing, the paradigm of cloud-edge collaboration has recently received extensive attention [4, 34]. DNN model partition is a typical example of the cloud-edge collaboration paradigm, which divides the giant model into different sub-models and deploys them on the cloud and the edge separately. However, the current DNN partition is mainly applied to model inference [12, 19] and considered individually for each edge site, which require an dedicated offline profiling phase to measure the network condition, the processing ability of the edge site, and the computing capacity of cloud server [43]. Federated learning is another type of collaboration paradigm to training DNN models on data distributed participants [16]. Meng et al. [24] proposed CNFGNN, which firstly utilizes the federated learning for spatial-temporal prediction. Specially, they utilize split learning and federated averaging to alternately optimize local temporal model and server-side spatial model. Federated learning, however, requires all participants to train a common model [11], which cannot be directly applied to edge workload forecasting since each edge site generally has a different number of workloads. Similar to us, He et al. [10] proposed Pyramid, a hierarchical spatial-temporal prediction framework. However, they consider intra-site correlations and inter-site correlations separately, whereas we jointly combine both relationships at each edge site to improve prediction accuracy.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose ELASTIC, which is the first study that leverages the cloud-edge collaborative paradigm for edge workload forecasting. We jointly consider the intra-site correlations among different workloads and the inter-site correlations. Comprehensive experiments on realistic edge workload traces confirm the effectiveness of ELASTIC. In the future, we plan to deploy the ELASTIC in production in the coming months and design an adjustable scheme adaptively pursuing the tradeoff between bandwidth consumption and prediction accuracy under different network conditions.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2020YFB1805502), NSFC (No.62032003, No.62102045), Beijing Nova Program (No.Z211100002121118), and CCF-ALIBABA (No.20220208). We hereby give special thanks to Alibaba Group for their contribution to this paper.

REFERENCES

[1] Cristiano Aguzzi, Lorenzo Gigli, Luca Sciullo, Angelo Trotta, and Marco Di Felice. 2020. From cloud to edge: Seamless software migration at the era of the web of things. *IEEE Access* 8 (2020), 228118–228135.

[2] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons.

[3] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the Symposium on Operating Systems Principles.* 153–167.

[4] Yuanrui Dong, Peng Zhao, Hanqiao Yu, Cong Zhao, and Shusen Yang. 2020. CDC: classification driven compression for bandwidth efficient edge-cloud collaborative deep learning. *arXiv preprint arXiv:2005.02177* (2020).

[5] Kan Guo, Yongli Hu, Zhen Qian, Hao Liu, Ke Zhang, Yanfeng Sun, Junbin Gao, and Baocai Yin. 2020. Optimized graph convolution recurrent neural network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* 22, 2 (2020), 1138–1149.

[6] Kan Guo, Yongli Hu, Zhen Sean Qian, Yanfeng Sun, Junbin Gao, and Baocai Yin. 2020. An optimized temporal-spatial gated graph convolution network for traffic forecasting. *IEEE Intelligent Transportation Systems Magazine* (2020).

[7] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. 2021. Hierarchical Graph Convolution Network for Traffic Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 151–159.

[8] Shengnan Guo, Youfang Lin, Ning Feng, and Chao Song. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 922–929.

[9] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, et al. 2020. Protean: VM allocation service at scale. In *USENIX Symposium on Operating Systems Design and Implementation.* 845–861.

[10] Qiang He, Zeqian Dong, Feifei Chen, Shuiguang Deng, Weifa Liang, and Yun Yang. 2022. Pyramid: enabling hierarchical neural networks with edge computing. In *Proceedings of the ACM Web Conference.* 1860–1870.

[11] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip learning as a decentralized alternative to federated learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems.* Springer, 74–90.

[12] Chuang Hu, Wei Bao, Dan Wang, and Fengming Liu. 2019. Dynamic adaptive DNN surgery for inference acceleration on the edge. In *Proceedings of the International Conference on Computer Communications.* 1423–1431.

[13] Changhee Joo and Ness B Shroff. 2017. A novel coupled queueing model to control traffic via QoS-aware collision pricing in cognitive radio networks. In *Proceedings of the International Conference on Computer Communications.* 1–9.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[17] Jitendra Kumar and Ashutosh Kumar Singh. 2021. Performance assessment of time series forecasting models for cloud datacenter networks' workload prediction. *Wireless Personal Communications* 116, 3 (2021), 1949–1969.

[18] Jitendra Kumar, Ashutosh Kumar Singh, and Rajkumar Buyya. 2021. Self directed learning based workload forecasting model for cloud resource management. *Information Sciences* 543 (2021), 345–366.

[19] En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. 2019. Edge AI: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications* 19, 1 (2019), 447–457.

[20] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[21] Haozhe Lin, Yushun Fan, Jia Zhang, and Bing Bai. 2021. Rest: Reciprocal framework for spatiotemporal-coupled predictions. In *Proceedings of the ACM Web Conference.* 3136–3145.

[22] Yuhua Lin and Haiying Shen. 2016. CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. *IEEE Transactions on Parallel and Distributed Systems* 28, 2 (2016), 431–445.

[23] Boyun Liu, Jingjing Guo, Chunlin Li, and Youlong Luo. 2020. Workload forecasting based elastic resource management in edge cloud. *Computers & Industrial Engineering* 139 (2020), 106136.

[24] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. 2021. Cross-node federated graph neural network for spatio-temporal data modeling. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 1202–1211.

[25] Chanh Nguyen, Cristian Klein, and Erik Elmroth. 2019. Multivariate LSTM-based location-aware workload prediction for edge data centers. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* 341–350.

[26] Amy Ousterhout, Joshua Fried, Jonathan Behrens, Adam Belay, and Hari Balakrishnan. 2019. Shenango: Achieving high CPU efficiency for latency-sensitive datacenter workloads. In *USENIX Symposium on Networked Systems Design and Implementation.* 361–378.

[27] Xiuquan Qiao, Pei Ren, Schahram Dustdar, Ling Liu, Huadong Ma, and Junliang Chen. 2019. Web AR: A promising future for mobile augmented reality—State of the art, challenges, and insights. *Proc. IEEE* 107, 4 (2019), 651–666.

[28] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49.

[29] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*. Springer, 362–373.

[30] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems* 28 (2015).

[31] Rachee Singh, Sharad Agarwal, Matt Calder, and Paramvir Bahl. 2021. Cost-effective cloud edge traffic engineering with Cascara. In *USENIX Symposium on Networked Systems Design and Implementation.* 201–216.

[32] Wei Sun and Xiaolong Xu. 2022. ALEDAR: An Attentions-based Encoder-Decoder and Autoregressive model for workload Forecasting of Cloud Data Center. In *IEEE International Conference on Computer Supported Cooperative Work in Design.* 59–64.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems* 27 (2014), 3104–3112.

[34] Shibo Wang, Shusen Yang, and Cong Zhao. 2020. Surveiledge: Real-time video query based on collaborative cloud-edge deep learning. In *Proceedings of International Conference on Computer Communications.* 2519–2528.

[35] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of the ACM Web Conference.* 1082–1092.

[36] Peter R Winters. 1960. Forecasting sales by exponentially weighted moving averages. *Management Science* (1960).

[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[38] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).

[39] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. 2021. From cloud to edge: a first look at public edge platforms. In *Proceedings of the ACM Internet Measurement Conference.* 37–53.

[40] Fanghua Ye, Zhiwei Lin, Chuan Chen, Zibin Zheng, and Hong Huang. 2021. Outlier-resilient web service QoS prediction. In *Proceedings of the ACM Web Conference.* 3099–3110.

[41] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[42] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).

[43] Letian Zhang, Lixing Chen, and Jie Xu. 2021. Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning. In *Proceedings of the ACM Web Conference.* 3111–3123.

[44] Zheng Zhang, Ming Zhang, Albert G Greenberg, Y Charlie Hu, Ratul Mahajan, and Blaine Christian. 2010. Optimizing Cost and Performance in Online Service Provider Networks.. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation.* 33–48.

[45] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 11106–11115.

[46] Yonghua Zhu, Weilin Zhang, Yihai Chen, and Honghao Gao. 2019. A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. *EURASIP Journal on Wireless Communications and Networking* 2019, 1 (2019), 1–18.

A TRAINING PIPELINE OF ELASTIC

In this section, we outline the detailed training pipeline of ELASTIC's global stage and local stage in Algorithm 1 and 2, respectively.

```
Algorithm 1: Training Pipeline of ELASTIC's Global Stage
Data: Historical observations of each edge site:
      $\{X^{t-P}, \dots, X^{t-1}\}$, global graphs, Length of the
      prediction period $H$
Result: Learned global spatial temporal model (GSTM) and
        each edge site's Aggregation Layer
Initialization;
for global training epoch $e_g = 1, 2, ..., E_g$ do
    for $\forall m \in \mathcal{M}$ in parallel do
        Pass both the input $X_m^{t-P:t-1}$ and target output
          $X_m^{t:t+H-1}$ through each site's Aggregation Layer;
        Send the results $Y_m^{t-P:t-1}$ and $Y_m^{t:t+H-1}$ to the cloud;
    end
    Concatenate the results of each edge site to get GSTM's
      training input $Y^{t-P:t-1}$ and target $Y^{t:t+H-1}$;
    Pass the input through GSTM to get prediction results
      $\widetilde{Y}^{t:t+H-1}$;
    Calculate the training loss for $Y^{t:t+H-1}$ and $\widetilde{Y}^{t:t+H-1}$
      using Eq.(5);
    Gradient backpropagation only in the direction of the
      input and send the corresponding gradient to each
      edge site's Aggregation Layer;
    Update the parameters of the GSTM and the
      Aggregation Layer of each edge site;
end
```

```
Algorithm 2: Training Pipeline of ELASTIC's Local Stage
Data: Historical observations of each edge site:
      $\{X^{t-P}, \dots, X^{t-1}\}$, local multi-view graphs, Length of
      the prediction period $H$
Result: Learned local spatial temporal model (LSTM) model
        and each edge site's Disaggregation Layer
Initialization;
for local training epoch $e_l = 1, 2, ..., E_l$ do
    for $\forall m \in \mathcal{M}$ in parallel do
        Pass the input $X_m^{t-P:t-1}$ through each site's already
          trained Aggregation Layer;
        Send the inference results $Y_m^{t-P:t-1}$ to the cloud and
          get the inference results of GSTM $\widetilde{Y}_m^{t:t+H-1}$;
        Fuse the output of the LSTM $\hat{X}_m^{t:t+H-1}$ and $\widetilde{Y}_m^{t:t+H-1}$
          to get the final prediction result $\widetilde{X}_m^{t:t+H-1}$;
        Calculate loss, backpropagate gradients and update
          the parameters of LSTM and Disaggregation Layer;
    end
end
```

B EVALUATION BASELINES IN DETAIL

The details of evaluation baselines are presented as follows:

(1) **ARIMA** [2]: It is a conventional time series prediction model that combines auto-regressive moving averages with integration.
(2) **LSTM** [30]: A type of recurrent neural network (RNN), which address long-term information preservation and short-term input skipping.
(3) **Graph WaveNet (a.k.a, GWNET)** [38]: One of the classic spatial-temporal forecasting methods utilizes dilated casual convolution and graph convolution to capture spatial-temporal dependencies.
(4) **DCRNN** [20]: One of the classic spatial-temporal forecasting methods captures the spatial dependency using bidirectional random walks and the temporal dependency using the encoder-decoder architecture with scheduled sampling.
(5) **ASTGCN** [8]: An attention-based spatial-temporal model consists of three independent components to respectively model complicated temporal properties of traffic flows.
(6) **GCRN** [29]: The graph convolutional recurrent network combines convolutional neural networks on graphs to identify spatial structures and RNN to find dynamic patterns.
(7) **STGCN** [41]: The spatial-temporal graph convolutional networks is built with complete convolutional structures instead of regular convolutional and recurrent units, which enable much faster training speed with fewer parameters.
(8) **HGCN** [7]: The hierarchical graph convolution network constructs a two-stream graph network to consider micro and macro traffic information.
(9) **OGCRNN** [5]: The optimized graph convolution recurrent neural network utilizes GCN to extract the spatial feature, GRU to extract the dynamic feature, and proposes an updating strategy to find an optimized graph matrix in a data-driven way in the training procedure.
(10) **OTSGGCNN** [6]: The optimized temporal-spatial gated graph convolution network captures the spatial-temporal traffic feature by an innovative graph convolution network with the graph constructed in a data-driven way.

The above baselines are all implemented by Pytorch 1.8.1. The only exception is ARIMA, which is implemented based on pmdarima [1]. The batch size is 64. The Adam Optimization is utilized. The original learning rate is 0.001, decaying with the ExponentialLR learning rate scheduler. The default dropout rate is 0.3. We train 50 epochs in the training phase. See [7] for more details.

C FURTHER DISCUSSION ON ELASTIC

C.1 Parameters Settings of Global Multi-view Graph.

As shown in Figure 10 and Figure 11, in order to control the sparsity of the global multi-view graph, the κ of spatial-view distacne graph is set to 400km, the κ of spatial-view RTT graph is set to 30ms. For temporal-view graph, the κ is set to 400 and $2 * 10^4$ for CPU and bandwidth datasets, respectively.

[1] http://alkaline-ml.com/pmdarima/

Table 5: Performance improvement of different local models applying the cloud-edge collaborative paradigm (with asterisk) on the average of MAE, SMAPE, and MSE over 12 prediction horizons.

	CPU			Bandwidth		
	Mean MAE	Mean SMAPE	Mean MSE	Mean MAE	Mean SMAPE	Mean MSE
LSTM	0.2523	34.51%	0.7288	12.048	64.01%	3757.15
LSTM*	**0.2053**	**26.21%**	**0.5726**	**11.201**	**52.93%**	**3379.15**
GRCN	0.2428	36.48%	0.7101	11.562	57.63%	3367.71
GRCN*	**0.2035**	**28.29%**	**0.5414**	**9.827**	**48.23%**	**2651.80**
STGCN	0.2119	27.17%	0.6473	6.345	44.59%	876.27
STGCN*	**0.2011**	**27.79%**	**0.5414**	**5.307**	**36.35%**	**696.23**
OTSGGCN	0.2049	26.38%	0.6066	5.398	36.26%	751.15
OTSGGCN*	**0.1927**	**24.59%**	**0.5276**	**5.200**	**34.50%**	**746.25**
ASTGCN	0.2137	30.60%	0.4935	12.566	69.79%	1431.38
ASTGCN*	**0.1698**	**23.62%**	**0.4314**	**6.581**	**45.36%**	**894.59**

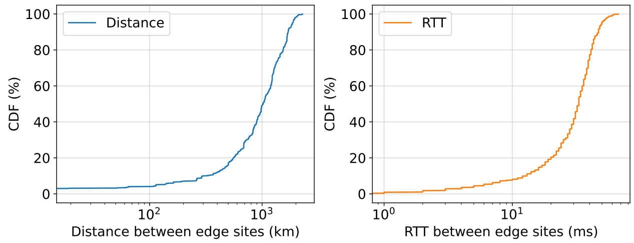

Figure 10: (a) CDF distribution of distance between edge sites; (b) CDF distribution of RTT between edge sites.

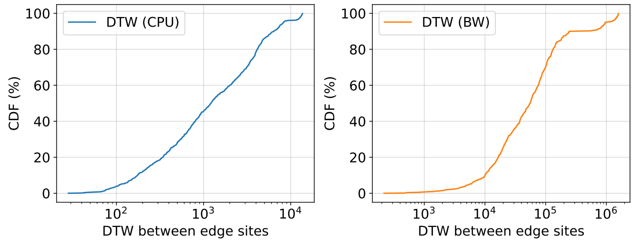

Figure 11: (a) CDF distribution of DTW distance between edge sites on CPU dataset; (b) CDF distribution of DTW distance between edge sites on BW dataset.

Specifically, we utilize some third-party (i.e., amap) open API [2] interfaces to query the distance between edge sites. Our dataset contains RTT records between edge sites every 5 minutes. We utilize the median of each pair and keep it constant.

C.2 Ablation Studies of Local Multi-view Graph (RQ2)

Similarly, we also conduct experiments with ELASTIC using five different adjacency matrix configurations to verify the effectiveness of our local multi-view graphs. Table 6 shows the average score of MAE, SMAPE, and MSE over 12 prediction horizons. We find that the adaptive-only model works best among identity, physical-only, and logical-only on all evaluation metrics. It indicates that even in the absence of a explict graph structure, self-adaptive adjacency matrices can produce good prediction performance. The phy-log-adaptive model achieves the lowest scores on all three evaluation metrics. It implies that if graph is provided, adding the self-adaptive adjacency matrix to the model could bring new and relevant information.

Table 6: Performance comparison of ELASTIC's local stage with different adjacency matrix configurations.

		Mean MAE	Mean SMAPE	Mean MSE
CPU	Identity	0.183	24.82%	0.498
	physical-only	0.170	22.46%	0.452
	logical-only	0.171	23.20%	0.444
	adaptive-only	0.162	22.96%	0.412
	phy-log -adaptive	**0.152**	**21.65%**	**0.400**
BW	Identity	5.04	40.60%	687.18
	physical-only	4.81	40.35%	637.39
	logical-only	4.83	38.58%	604.48
	adaptive-only	4.68	38.04%	592.23
	phy-log -adaptive	**4.52**	**37.03%**	**588.27**

C.3 Apply to other Prediction Models (RQ3)

To further verify the effectiveness of ELASTIC's cloud-edge collaboration framework, we apply it to other local prediction models. Table 5 shows the average score of MAE, SMAPE, and MSE over 12 prediction horizons on both CPU and bandwidth datasets. The models marked with an asterisk apply our cloud-edge collaboration framework, whereas those without are original models. According to the table, after adding the cloud-edge collaboration framework, all models' prediction performance has improved significantly compared with the original. LSTM has the most significant improvement on the CPU dataset, while ASTGCN has the most significant improvement on the bandwidth dataset.

[2] https://lbs.amap.com/