Pyramid: Enabling Hierarchical Neural Networks with Edge Computing

Qiang He[1], Zeqian Dong[1], Feifei Chen[2], Shuiguang Den[3], Weifa Liang[4], Yun Yang[1]
[1]Department of Computing Technologies, Swinburne University of Technology, Australia
[2]School of Information Technology, Deakin University, Australia
[3]College of Computer Science and Technology, Zhejiang University, China
[4]Department of Computer Science, City University of Hong Kong, China
qhe,zdong,yyang@swin.edu.au,feifei.chen@deakin.edu.au,dengsg@zju.edu.cn,weifa.liang@cityu.edu.hk

ABSTRACT

Machine learning (ML) is powering a rapidly-increasing number of web applications. As a crucial part of 5G, edge computing facilitates edge artificial intelligence (AI) by ML model training and inference at the network edge on edge servers. Compared with centralized cloud AI, edge AI enables low-latency ML inference which is critical to many delay-sensitive web applications, e.g., web AR/VR, web gaming and Web-of-Things applications. Existing studies of edge AI focused on resource and performance optimization in training and inference, leveraging edge computing merely as a tool to accelerate training and inference processes. However, the unique ability of edge computing to process data with context awareness, a powerful feature for building the web-of-things for smart cities, has not been properly explored. In this paper, we propose a novel framework named Pyramid that unleashes the potential of edge AI by facilitating homogeneous and heterogeneous hierarchical ML inferences. We motivate and present Pyramid with traffic prediction as an illustrative example, and evaluate it through extensive experiments conducted on two real-world datasets. The results demonstrate the superior performance of Pyramid neural networks in hierarchical traffic prediction and weather analysis.

CCS CONCEPTS

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

KEYWORDS

Web of Things, edge AI, machine learning, edge computing

ACM Reference Format:
Qiang He[1], Zeqian Dong[1], Feifei Chen[2], Shuiguang Den[3], Weifa Liang[4], Yun Yang[1]. 2022. Pyramid: Enabling Hierarchical Neural Networks with Edge Computing. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3511990

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511990

1 INTRODUCTION

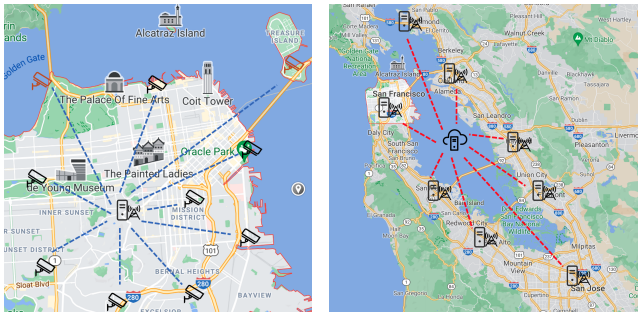
The last decade has witnessed the success of artificial intelligence (AI) powered by machine learning (ML), especially deep learning that has been widely employed to transform our lives, from face recognition and natural language processing to medical diagnosis [25]. In recent years, the proliferation of mobile and Web-of-Things (WoT) applications has profoundly increased the volume of data generated at the network edge. These data are normally transmitted to the cloud to facilitate centralized ML model training and inference. This cloud AI model suffers from inherent limitations. First, it struggles to ensure low latency for delay-sensitive applications, e.g., web AR/VR [43], web gaming [32] and many WoT applications [40]. For example, offloading a camera frame to an AWS server and performing a computer vision task take more than 200 milliseconds end-to-end [4]. Second, it incurs excessive traffic over the backhaul network that is already under tremendous pressure.

Fortunately, edge computing offers a new computing paradigm that enables ML at the network edge to overcome the above limitations. In an edge computing environment, edge servers equipped with computing resources are deployed at base stations or access points [16]. ML models can be trained and deployed on edge servers instead of remote cloud servers to facilitate *edge AI* [44]. Compared with cloud AI, edge AI offers the following benefits:

- Model training and inference can be performed within close proximity to data and end-devices. This enables real-time interactions between AI applications and end-devices.
- Model training and inference at the network edge can minimize the traffic produced over the backhaul network.
- Edge servers' computing and storage resources suffice to facilitate ML model training and inference that are too large and/or too complex for lightweight end-devices.
- Powered by 5G, the high-speed links between end-devices and edge servers minimize the delays caused in ML model training and inference.

In an edge computing environment, an edge server communicates with end-devices within its coverage area [16]. Data collected by these end-devices, e.g., cameras, sensors and mobile phones, can be transmitted to the edge server for processing. On one hand, an ML model deployed on the edge server can capture the *intra-level* spatial-temporal correlations in the data. Take traffic prediction as an example. The road traffics in a region can be predicted based on the traffic data collected by the sensors deployed in the region, as demonstrated in Fig. 1(a). This is referred to as *local prediction*. In recent years, a series of graph neural networks have been proposed to

make traffic predictions based on such intra-region spatial-temporal correlations [31, 41, 50]. On the other hand, an ML model can be deployed on a remote cloud server to capture the *inter-level* spatial-temporal correlations in the data collected by multiple edge servers. For example, the traffics on a freeway connecting multiple regions can be predicted based on the traffic data collected by the edge servers deployed in these regions, as demonstrated in Fig. 1(b). This is referred to as *global prediction*.



(a) Local Traffic Prediction (b) Global Traffic Prediction

Figure 1: Hierarchical Traffic Prediction

The benefits brought by edge computing enable *hierarchical ML inference* that allows AI applications to perform both local and global inferences. This is unprecedented prior to edge computing. In this paper, we propose Pyramid, a novel framework that facilitates hierarchical ML inference in the edge computing environment. Based on the Pyramid framework, a *Pyramid neural network* (NN) can be deployed across the edge and the cloud, as demonstrated in Fig. 2. Training data collected by end-devices is fed to the Pyramid NN. A properly-trained Pyramid NN can make local predictions on edge servers and global predictions on the cloud server. Compared with state-of-the-art studies [11, 31, 41, 50, 59], the unique features of Pyramid are summarized as follows.

- Pyramid NNs can make homogeneous and heterogeneous hierarchical predictions while existing spatial-temporal graph neural networks (GNNs) enable local predictions only. This unleashes many new possibilities for building the WoT for smart cities, e.g., smart transportation management, weather analysis, energy consumption management, etc.
- Pyramid NNs capture the intra-level spatial-temporal correlations at the local level, as well as the inter-level spatial-temporal correlations between the local level and the global level. Compared with conventional spatial-temporal GNNs, Pyramid NNs obtain higher inference accuracy and faster convergence.
- Pyramid NNs can make local predictions on edge servers with low inference delays. This contributes to real-time smart city management and allows end-devices like autonomous vehicles to respond rapidly to environmental changes like traffic congestion and accidents.

In addition to traffic prediction, Pyramid also supports other ML applications, e.g., air quality prediction [13], energy consumption prediction [10], etc. In the remainder of this paper, we present and discuss Pyramid with its application to traffic prediction, a key and active task in smart city management [31, 50].

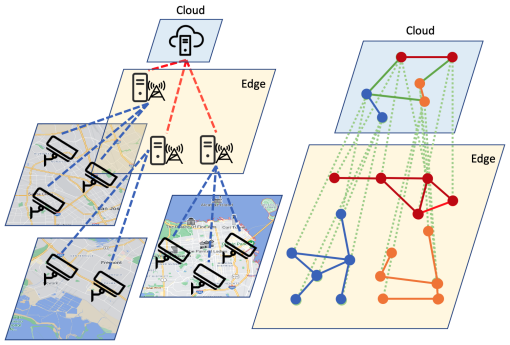


Figure 2: Pyramid in the Edge Computing Environment

2 PROBLEM STATEMENT

In this section, we introduce definitions and formulate the problem of hierarchical traffic prediction. A summary of the key notations used in this paper is given in Table 1.

Table 1: Notations

Notation	Definition
A_k^{rr}	Road-road adjacency matrix for kth region
A_l^{rf}	Road-freeway adjacency matrix for lth freeway
A^{ff}	Freeway-freeway adjacency matrix
K	Number of regions
L	Number of freeways
M	Number of all freeway sensors
N_k	Number of road sensors in the $k$the region
N	Number of road sensors in all regions
V_k^r	Road sensors monitoring roads in kth region
V_l^f	Freeway sensors monitoring lth freeway
$\mathcal{X}$	Series of traffic flow data
$\mathcal{X}_k^r$	Road traffic matrix for kth region

By making use of the traces of historical traffic data collected by sensors, a Pyramid NN can learn spatial-temporal traffic correlations by 1) predicting traffics on the roads in individual regions; and 2) predicting traffics on the freeways connecting these regions.

Traffic Network. Given K regions connected by L freeways, the N_k *road sensors* monitoring the roads in the kth ($1 \le k \le K$) region are represented as $V_k^r = \{v_{k,1}^r, \cdots, v_{k,N_k}^r\}$. The set of the N road sensors are represented as $V^r = \{V_1^r, \cdots, V_K^r\}$. On each of the L freeways, there are usually at least three *freeway sensors*, including two endpoint freeway sensors, one at each end that connects the freeway to one of the two corresponding regions, and a midpoint freeway sensor in the middle of the freeway. The sensors monitoring the lth freeway are represented as $V_l^f = \{v_{l,1}^f, v_{l,2}^f, v_{l,3}^f\}$. In total, there are $M(M = 3L)$ freeway sensors, represented by a set $V^f = \{V_1^f, \cdots, V_L^f\}$.

Adjacency Matrices. The spatial distance between roads plays a crucial role in road traffic correlations [11, 20, 50]. In an individual region, the traffic condition at a location impacts its adjacent locations significantly [50]. For example, a congested road can

quickly reduce the traffic flow on nearby roads. Thus, similar to [11, 20, 50], a road-road adjacency matrix for each of the K regions based on the spatial distance between roads is constructed, denoted by $A_k^{rr} \in \mathbb{R}^{N_k \times N_k}$ that corresponds to the kth region. On the freeway that connects two regions, say the k_1th and the k_2th regions, the traffic condition at an endpoint is impacted significantly by the traffic conditions on adjacent roads in the corresponding region. The traffic condition at the midpoint is impacted by the road traffics in both regions. Their connections are described by a road-freeway adjacency matrix $A_l^{rf} \in \mathbb{R}^{(N_{k_1}+N_{k_2})\times 3}$. The traffic conditions on connected freeways are also correlated. We use a matrix A^{ff} to describe their connections.

Problem Definition. The historical traffic data collected by a road sensor or a freeway sensor v over a period of T is represented by a sequence $\mathcal{X}_v = (x_{v,1}, x_{v,2}, \cdots, x_{v,T}) \in \mathbb{R}^T$. The historical traffic data collected by all the road and freeway sensors are represented as $\mathcal{X} = (\mathcal{X}_1, \cdots, \mathcal{X}_N, \mathcal{X}_{N+1}, \cdots, \mathcal{X}_{N+M}) \in \mathbb{R}^{(N+M)\times T}$. Given $\mathcal{X}$, hierarchical traffic prediction aims to 1) predict the traffic conditions on the roads in individual regions at a future time step $T+1$, denoted as $\mathcal{X}_{T+1}^r \in \mathbb{R}^N$; and 2) predict the traffic conditions on the freeways connecting individual regions at a future time step $T+1$, denoted as $\mathcal{X}_{T+1}^f \in \mathbb{R}^M$.

3 PYRAMID NEURAL NETWORK

Fig. 3 overviews the Pyramid NN built for hierarchical traffic predictions. It is comprised of two main components: 1) Local Learners deployed on edge servers for predicting road traffics in corresponding regions; and 2) Global Learner deployed on a cloud server for predicting traffic conditions on freeways connecting individual regions. First, the traffic data collected by road sensors in a region are transmitted to the edge server covering that region. The data is then fed to the Local Learner to capture spatial-temporal road traffic correlations for predicting road traffics. Then, the edge servers covering different regions send the extracted road traffic features to the cloud server to feed the Global Learner. The Global Learner will aggregate these features with the data collected by freeway sensors to capture the spatial-temporal freeway correlations for predicting freeway traffics.

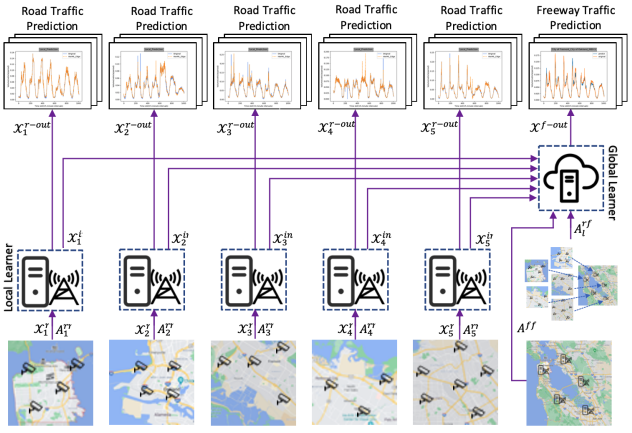


Figure 3: Hierarchical Traffic Prediction based on Pyramid

3.1 Local Prediction

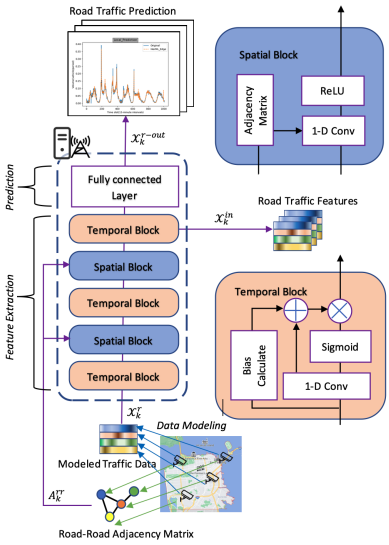


Figure 4: Local Learner (for Road Traffic Prediction)

Based on Pyramid, a Local Learner is deployed on each edge server to predict road traffics in the region covered by the edge server. It is comprised of three main components, i.e., data modeling, feature extraction, and prediction components, as shown in Fig. 4. The key is to capture complex spatial-temporal road traffic correlations. State-of-the-art approaches, like STRN [31], STGNN [50], LSGCN [20], and HGCN [11], have adopted a methodology similar to ST-GCN [56], i.e., to capture spatial correlations with graph convolutional networks (GCNs) and temporal correlations with gated linear units (GLUs), or gate recurrent units (GRUs). Many approaches, e.g., ST-GCN, LSGCN and HGCN employ the "sandwich" structure that consists of two temporal blocks and a spatial block in-between. In particular, ST-GCN and HGCN stack multiple such sandwiches to capture complex spatial-temporal correlations. However, the temporal blocks between stacked sandwiches incur extra training overheads and inference delays but do not necessarily increase inference accuracy. Local Learner employs a novel "double decker sandwich" structure that consists of three temporal blocks interleaved with two spatial blocks to capture spatial-temporal road traffic correlations.

3.1.1 Data Modeling. We put the historic traffic data collected by road sensors into matrices readable to Local Learner. Let us assume that the historical traffic data in the kth region are collected by the N_k road sensors in the region over T time steps. The input matrix $\mathcal{X}_k^r \in \mathbb{R}^{N_k \times T}$ for the corresponding Local Learner can be represented as follows.

$$\mathcal{X}_k^r = \begin{bmatrix} x_{1,1}^r & \cdots & x_{1,T-1}^r & x_{1,T}^r \\ x_{2,1}^r & \cdots & x_{2,T-1}^r & x_{2,T}^r \\ \vdots & \ddots & \vdots & \vdots \\ x_{N_k-1,1}^r & \cdots & x_{N_k-1,T-1}^r & x_{N_k-1,T}^r \\ x_{N_k,1}^r & \cdots & x_{N_k,T-1}^r & x_{N_k,T}^r \end{bmatrix} \quad (1)$$

3.1.2 Feature Extraction. Local Learner employs temporal blocks to extract temporal road traffic features from the input matrix $\mathcal{X}_k^r$. A temporal block includes a modified gate linear unit (GLU) [8]. The advantage of using GLU in the temporal block is that the gate unit transmits both linear and non-linear features to subsequent layers. This will retain most information for inference during the forwarding propagation. Eq. (2) shows the operation performed by the GLU to learn parameters $W \in \mathbb{R}^{N_k}$ and $V \in \mathbb{R}^{N_k}$:

$$\mathcal{X}^{t-out} = (W\mathcal{X}^{t-in} + b) \otimes \sigma(V\mathcal{X}^{t-in} + c) \tag{2}$$

where $b, c \in \mathbb{R}^{N_k}$ are the biases for the direct connect component and the activation component, and σ is a sigmoid activation layer. Specifically, the convolution for both components is calculated first. Then, the sigmoid activation layer filters the activating part of the convolution before the two parts are multiplied. There are three temporal blocks in Local Learner. The input to the first temporal block is matrix $\mathcal{X}_k^r$ produced by the data modeling block, i.e., $\mathcal{X}_{in}^t = \mathcal{X}_k^r$. The inputs to the other two temporal blocks are the matrices produced by their precedent spatial blocks.

Local Learner employs two spatial blocks to extract spatial road traffic features. The main component of a spatial block is a Graph Convolutional Network (GCN) [24], which captures spatial road traffic correlations through graph convolution. Eq. (3) shows the operation performed by the GCN to learn parameter $W \in \mathbb{R}^{N_k}$:

$$\mathcal{X}^{s-out} = \tau(W\mathcal{X}^{s-in} \cdot D^{-1/2}\hat{A}D^{-1/2}) \tag{3}$$

where $\mathcal{X}^{s-in}$ is the output of the precedent temporal block (see Eq. (2)), i.e., $\mathcal{X}^{s-in} = \mathcal{X}^{t-out}$, τ is a ReLU activation layer, D is the diagonal degree matrix with $D_{i,i} = \sum_j \hat{A}_{i,j}$, $\hat{A}_{i,j} = A_{i,j} + I_i$, and A is the adjacency matrix. When used by Local Learner, A is the road-road adjacency matrix A_k^{rr} defined in Section 2.

3.1.3 Prediction. Local Learner aims to predict $\mathcal{X}_{T+1}^r \in \mathbb{R}^{N_k}$. It employs a fully connected layer to do so, taking all the spatial-temporal road traffic features outputted by the feature extraction component as input:

$$\mathcal{X}^{r-out} = W\mathcal{X}^{in} + b \tag{4}$$

where $W \in \mathbb{R}^{fs \times 1}$ is the trainable weight and fs is the size of the extracted features. Making local predictions, the output of a Local Learner is the road traffics at time step $T+1$, i.e., $\mathcal{X}^{r-out} = \mathcal{X}_{T+1}^r$. In the meantime, it transmits $\mathcal{X}^{in}$ to Global Learner deployed in the cloud to facilitate global predictions, i.e., freeway traffic predictions.

3.2 Global Prediction

The traffic on a freeway is spatially and temporally correlated with the traffics in the regions connected by the freeway as well as those on other freeways. Based on the Pyramid framework, a Global Learner is deployed in the cloud to capture these freeway traffic correlations. It is comprised of two main components, i.e., feature extraction and prediction components, as shown in Fig. 5.

3.2.1 Feature Extraction. The inputs to Global Learner are the road traffic features extracted by Local Learners deployed on edge servers in different regions:

$$\mathcal{X}^r = concat_{k=1,\cdots,K}(\mathcal{X}_k^{in}) \tag{5}$$

where $\mathcal{X}_k^{in}$ is transmitted from the kth Local Learner.

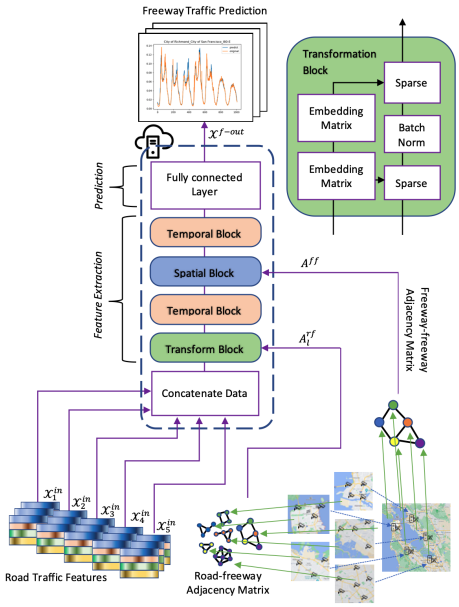


Figure 5: Global Learner (for Freeway Traffic Prediction)

From $\mathcal{X}^r$, Global Learner captures three types of freeway traffic correlations: 1) the spatial correlations between road traffics and freeway traffics; 2) the spatial traffic correlations between freeways; and 3) the temporal correlations in freeway traffics.

Global Learner employs a novel transformation block to convert road traffic features $\mathcal{X}^r$ into freeway traffic features $\mathcal{X}^f$. It uses two sparse layers and a batch normalization layer in-between to capture spatial correlations between freeway traffics and road traffics. The first sparse layer converts the N-dimensional road traffic features $\mathcal{X}^r$ to Q-dimensional intermediate features:

$$\mathcal{X}^{inter} = (E_0 \odot W_0) * \mathcal{X}^r + b_0 \tag{6}$$

where W_0 is the parameter to learn, b_0 is a random bias commonly used in neural network training, E_0 is the embedding matrix for the first sparse layer. The second sparse layer converts intermediate features $\mathcal{X}^{inter}$ to M-dimensional freeway traffic features:

$$\mathcal{X}^f = (E_1 \odot W_1) * \mathcal{X}^{bn-inter} + b_1 \tag{7}$$

where W_1 is the parameter to learn, b_1 is a random bias, $\mathcal{X}^{bn-inter}$ is the intermediate features normalized by the batch normalization layer (to be discussed later), E_1 is the embedding matrix for the second sparse layer.

Embedding layers E_0 and E_1 are constructed based on the road-freeway adjacency matrices $A_l^{rf}, l = 1, \cdots, L$. First, a $N \times M$ global road-freeway adjacency matrix A^{rf} is built from $A_l^{rf}, l = 1, \cdots, L$, where each binary element $a_{i,j} \in A^{rf}$ indicates whether the ith road sensor is relevant to the jth freeway sensor. Then, A^{rf} factorized into $N \times Q$ E_0 and $Q \times M$ E_1 for the first and the second sparse layers in the transformation block, where $N > Q > M$ and $Q \mod M = 0$.

The batch normalization layer [21] between the sparse layers aims to reduce the internal covariate shift. It stabilizes the training of Global Learner by fixing the mean and variance of $\mathcal{X}^{inter}$. First,

it calculates the mean and variance of the elements in $\mathcal{X}^{inter}$:

$$\mu = \frac{1}{Q}\sum_{q=1}^{Q}\mathcal{X}^{inter} \quad \sigma^2 = \frac{1}{Q}\sum_{q=1}^{Q}(\mathcal{X}^{inter} - \mu)^2 \tag{8}$$

It then normalizes each dimension of $\mathcal{X}^{inter}$:

$$\mathcal{X}^{bn-inter} = \frac{\mathcal{X}^{inter} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{9}$$

where ϵ is a minimum constant for numerical stability [21].

By learning W_0 (Eq. (6)) and W_1 (Eq. (7)), Global Learner captures the spatial correlations between road traffics and freeway traffics. Next, given $\mathcal{X}_f$ (Eq. (7)) from the transformation layer, Global Learner employs a "sandwich" structure to capture the spatial-temporal freeway traffic correlations. Unlike Local Learners, its spatial block uses A^{ff} instead of A_k^{rr} in Eq. (3).

3.2.2 Prediction. Global Learner employs a fully-connected layer to produce the prediction results:

$$\mathcal{X}^{f-out} = W\mathcal{X}^{f} + b \tag{10}$$

where W is the parameter to learn and b is a random bias. Making global predictions, the output of Global Learner is the freeway traffics at time step $T+1$, i.e., $T^{f-out} = X_{T+1}^{f} \in \mathbb{M}^M$.

4 EVALUATION

To validate the applicability and performance of Pyramid, we implement, train and test two Pyramid NNs, Pyramid-TP for traffic prediction and Pyramid-WA for weather analysis. Due to the space limit, this section will focus on Pyramid-TP and the evaluation of Pyramid-WA is presented in Appendix A.

4.1 Experiment Setup

Dataset. Pyramid-TP is trained on the PeMSD4 dataset [3] widely used for traffic predictions [1, 12, 20, 56]. It contains the traffic flow information in five regions surrounding the San Francisco Bay Area, collected by 963 road sensors and 57 freeway sensors over 7 months between January and August in 2018.

Environment. To simulate a cloud-edge environment for experiments, we hire a g3s.xlarge Amazon EC2 instance as the cloud server and deploy five virtual machines in our private data center as edge servers, each with a 4-core processor, 16GB RAM and a GPU comparable to GeForce RTX 2080 Ti. We run network tests in this environment and find that the network latency between the cloud server and edge servers is between 90 and 140 milliseconds. The network latency between sensors and edge servers is negligible because 5G technologies ensure ultra-low latencies between end-devices and edge servers. In traffic prediction, the uplink speed plays an important role in communication performance. Thus, similar to [58], we throttle edge servers' maximum uplink speed at 50MB/s, 500MB/s and 1,000MB/s to simulate slow, medium and fast networking environments.

Baselines. In the experiments, Pyramid-TP is evaluated against five representative traffic prediction approaches, two statistical approaches, including ARIMA and SVR, and four neural network (NN) based approaches, including LSTM, GRU, DDNN and HGCN.

1. **ARIMA** [51]: Auto Regression Integration Moving Average is a widely-used approach for time series analysis. It is a common baseline in traffic predictions [11, 31, 50, 55, 56] and predicts road traffic and freeway traffic individually without considering their spatial correlations.
2. **SVR** [46]: Support Victor Regression uses a support vector machine to perform regression analysis for traffic predictions. Its main limitations are its stability assumptions and the lack of consideration in spatial-temporal traffic correlations. It is also a common baseline in traffic predictions [50, 56].
3. **LSTM** [47]: Long-Short Term Memory is a classic NN model for analyzing traffic time series [11, 31, 50, 56]. It is incapable of modeling spatial traffic correlations.
4. **GRU** [7]: Gated Recurrent Unit is a lightweight variant of LSTM with gate units and is often employed to analyze traffic sequences as time series [11, 20, 56]. Similar to LSTM, it does not model spatial traffic correlations.
5. **DDNN** [49]: Deep Distributed Neural Network is a hierarchical NN model that can be deployed across the edge and the cloud. Its key idea is to partition a convolutional neural network (CNN) into two parts to be deployed at the edge and in the cloud. However, it processes the traffic data collected by different sensors as individual traffic sequences and does not model the spatial-temporal correlations in and between them.
6. **HGCN** [11]: Hierarchical Graph Convolution Network is the state-of-the-art approach for traffic predictions, outperforming ST-GCN [56] and GWNET [54]. It models the temporal correlations within individual traffic sequences and the spatial correlations between them.

Experiment Settings. Following the same settings as [31, 50], the PsMSD4 dataset is split into three parts, the first part containing 70% of the data for training, the second part containing 20% for test, and the third part containing 10% for validation. There are a total of 32,256 time steps in the dataset. The 10-minute traffic prediction at a time step is made based on the previous 20 time steps. ARIMA, SVR, LSTM, GRU and DDNN make predictions for individual locations, one per sensor, using the traffic time series collected by these sensors as individual traffic sequences. HGCN models the spatial-temporal correlations in the traffic data collected by all sensors, including road sensors and freeway sensors, with one graph neural network.

All the baselines are implemented and trained based on open-source projects, following default hyper-parameter settings from corresponding papers or common settings. When testing LSTM and GRU, we train two models on the cloud server, one for predicting road traffics solely based on road traffics and the other for predicting freeway traffics solely based on freeway traffics. When testing DDNN, we train and deploy multiple models across the edge and the cloud, one for each region with two exits. The exits at the edge make road traffic predictions and the ones in the cloud make freeway traffic predictions. When testing HGCN, we train one model on the cloud server for predicting road traffics and freeway traffics, treating them equally. When testing Pyramid-TP, we train one model with exits across the edge and the cloud, the ones at the edge for road traffic predictions and the one on the cloud server for freeway traffic predictions, similar to DDNN and BranchyNet [48].

Pyramid-TP is configured with 16 features for temporal blocks and 8 features for spatial blocks. We employ the Adam optimizer [23] for model training, with a learning rate of 0.0001, a batch size of 1,008, and the mean square error (MSE) as the loss. We run each approach five times and report the average results.

Performance Metrics. Prediction accuracy is measured by four commonly-used performance metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-Squared (R^2). Lower MSE, MAE, RMSE and higher R-Squared values indicate higher accuracy.

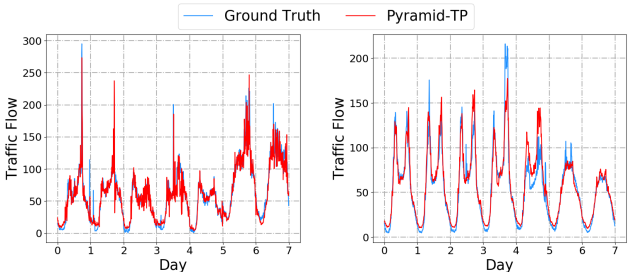
4.2 Experimental Results

Overall Performance. The results are shown in Table 2 where the best results are underlined. The performance of ARIMA and SVR is impacted by the window size, i.e., the number of previous time steps taken as input to make predictions. We demonstrate their results obtained with windows size = 25 and 10. The performance of neural networks is impacted by the number of training epochs. We show their results obtained with 500 and 1,000 epochs. The following key observations can be derived from Table 2.

1. Statistical approaches like ARIMA and SVR suffer poor performance. A larger window size takes more data into consideration but lowers their accuracy in general. This indicates that these approaches cannot handle the temporal dynamics in traffic sequences well. The large accuracy margins between them and NN-based approaches tell us that they are incapable of modeling complex spatial-temporal traffic correlations.
2. Taking the advantages offered by the RNN architecture in analyzing time series, LSTM and GRU achieve remarkably higher accuracy than statistical approaches. This indicates the importance of modeling temporal traffic correlations. However, LSTM and GRU both overlook spatial traffic correlations.
3. DDNN has the worst performance among all five NN-based approaches. In certain cases, it is even outperformed by ARIMA and SVR. This shows that DDNN is not capable of modeling complex spatial-temporal traffic correlations.
4. HGCN, as the state-of-the-art approach for traffic prediction, improves prediction accuracy further compared with LSTM and GRU by modeling both spatial and temporal traffic correlations.
5. Pyramid-TP achieves the highest prediction accuracy in almost all the cases. With 1,000 epochs, it outperforms all the other approaches significantly. This comes from its outstanding ability to model spatial-temporal road traffic correlations, and three types of spatial-temporal freeway traffic correlations discussed in Section 3.2.1.

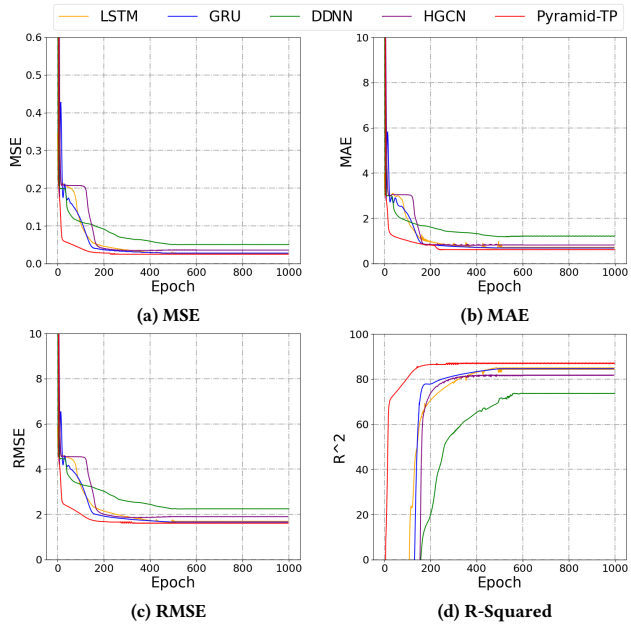
Model Fitting. Fig. 6 evaluates the ground truth against the predictions by Pyramid-TP (with 500 epochs) for a road and a freeway randomly selected over seven days. We can see that it fit in the ground truth perfectly, except only a few abrupt changes in ground-truth traffics. This demonstrates the ability of Pyramid-TP to handle complex traffics over time.

Model Convergence. Fig. 7 visualizes the convergence of the five NN models for local prediction. Pyramid-TP is always the first to converge, taking about 150 epochs. Unlike HGCN that attempts to capture traffic correlations with one neural network, Pyramid-TP deploys a Local Learner on each edge server to capture the traffic correlations in individual regions. These Local Learners are trained in parallel, which accelerates model convergence for local predictions. In the meantime, we can see that Pyramid-TP achieves the highest prediction accuracy after convergence. This implies that the local traffics in different regions are not strongly correlated and road traffic correlations can be captured accurately by these Local Learners rapidly and accurately. This also validates the ability of Pyramid to capture intra-level spatial-temporal correlations accurately and rapidly.



(a) Local Traffic Prediction (b) Global Traffic Prediction

Figure 6: Predictions over Time



(a) MSE (b) MAE

(c) RMSE (d) R-Squared

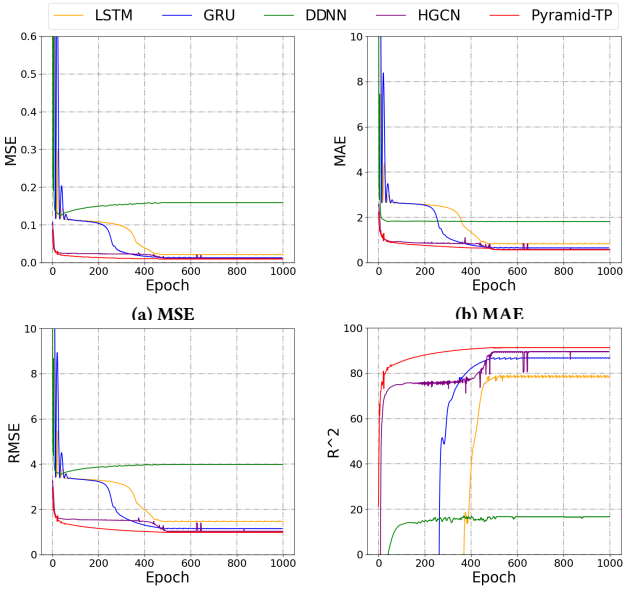
Figure 7: Model Convergence for Local Prediction

Fig. 8 visualizes the convergence of the five NN models for global prediction. Again, Pyramid-TP is always the first to converge, slightly faster than HGCN and significantly than the others. Pyramid-TP and HGCN's fast convergence indicates that the combination of GNN and RNN is capable of capturing spatial-temporal traffic correlations rapidly. Similar to Pyramid-TP, HGCN also captures the traffic correlations between different regions, which contributes to freeway traffic predictions. This allows it to converge almost as fast as Pyramid-TP. However, it treats road traffics and freeway traffics equally and fails to capture the multiple types of spatial-temporal freeway traffic correlations. Thus, its accuracy

Table 2: Prediction Accuracy

	Local Prediction (Road Traffic)								Global Prediction (Freeway Traffic)							
	MSE	MAE	RMSE	R-Sqaured	MSE	MAE	RMSE	R-Sqaured	MSE	MAE	RMSE	R-Sqaured	MSE	MAE	RMSE	R-Sqaured
Statistical	Window Size = 25				Window Size = 10				Window Size = 25				Window Size = 10			
ARIMA [51]	0.09	2.71	4.47	32.40%	0.07	1.96	3.71	45.71%	0.10	2.00	2.96	25.88%	0.07	1.22	2.09	47.25%
SVR [46]	0.09	2.09	2.88	24.79%	1.18	1.88	0.05	69.89%	0.09	1.62	2.21	16.90%	0.07	0.77	1.26	46.83%
NN-based	500 Epochs				1,000 Epochs				500 Epochs				1,000 Epochs			
LSTM [47]	0.03	0.72	1.66	84.32%	0.03	0.64	1.36	87.32%	0.02	0.83	1.45	77.98%	0.01	0.73	1.39	80.12%
GRU [7]	0.02	0.92	1.66	84.61%	0.02	0.64	1.21	89.66%	0.01	0.45	0.96	89.48%	0.01	0.44	0.75	91.63%
DDNN [49]	0.05	1.23	2.25	72.60%	0.05	0.99	2.06	79.61%	0.16	1.81	3.99	40.31%	0.18	1.63	2.30	45.53%
HGCN [11]	0.03	0.84	1.91	81.37%	0.02	0.74	1.59	84.34%	0.01	0.58	0.83	90.45%	0.01	0.63	1.07	90.15%
Pyramid-TP	0.02	0.79	1.54	85.98%	0.01	0.25	0.46	92.81%	0.01	0.67	1.01	90.92%	0.01	0.40	0.63	95.46%

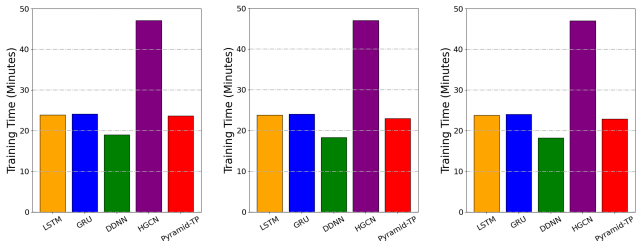
after convergence is much lower than Pyramid-TP. The results indicate that Pyramid can capture inter-level spatial-temporal correlations accurately and rapidly.



(a) MSE (b) MAE (c) RMSE (d) R-Squared

Figure 8: Model Convergence for Global Prediction

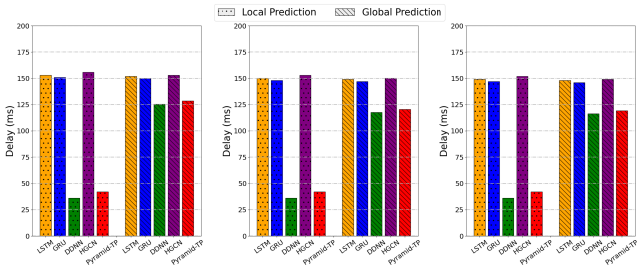
Training Time. Fig. 9 compares the five approaches' training times over 500 epochs in slow, medium and fast networking environments. Overall, HGCN takes much more time than the other four approaches to complete and Pyramid-TP's training time is comparable to LSTM, GRU and DDNN overall. When the uplink speed is 50MB/s, Pyramid-TP's training time is almost the same as LSTM and GRU, as shown in Fig. 9(a). When the uplink speed increases to 500MB/s, Pyramid-TP takes less time to complete the training process than LSTM and GRU, as shown in Fig. 9(b). Another increase in the uplink speed to 1,000MB/s does not reduce Pyramid-TP's training time further, as shown in Fig. 9(c). When the uplink speed is above 500MB/s, the data transmission time between edge servers and the cloud server is no longer a performance bottleneck. DDNN also benefits from a high uplink speed, taking an average of only 22.8 minutes to complete across all three cases, 11.2% less than Pyramid-TP. Its simple model is the root cause for this advantage. However, it is also the reason for its poor prediction accuracy, as demonstrated before.



(a) Slow (50MB/s) (b) Medium (500MB/s) (c) Fast (1000MB/s)

Figure 9: Training Time Comparison

Inference Delay. Compared with the cloud AI, a main advantage offered by edge AI is low inference delay [58]. Fig. 10 compares the average inference delays achieved by the five NN-based approaches. Being able to perform local inferences at the edge, Pyramid-TP and DDNN outperform the other approaches profoundly in making fast road traffic predictions, taking only an average of only 36 milliseconds and 42 milliseconds, respectively. When it comes to global prediction, Pyramid-TP and DDNN's inference delays are impacted by the uplink speed, similar to what we observed in Fig. 9. Overall, Pyramid-TP and DDNN make the fastest highway traffic predictions. An increase in the uplink speed broaders their advantages over LSTM, GRU and HGCN. This indicates the high efficiency of Pyramid-TP in making global predictions.



(a) Slow (50MB/s) (b) Medium (500MB/s) (c) Fast (1000MB/s)

Figure 10: Inference Time Comparison

5 RELATED WORK

Decentralized Machine Learning. To overcome the limitations of centralized machine learning (ML), many new ML technologies have been developed in the last several years. For example, data parallelism [2, 27, 29] and model parallelism [9, 26] have been proposed to accelerate the training of ML models, especially large-scale ML models. The former partitions a training dataset into multiple subsets, one for each worker that trains a replica of the ML model. The latter partitions an ML model into multiple parts to be trained by multiple workers in parallel. While these efforts are devoted to decentralizing model training, some researchers have attempted to reduce inference delays by enabling decentralized model inference at locations close to data sources and users. A variety of techniques have been proposed to enable mobile AI by performing ML model training and inferences on mobile devices [36, 53, 60]. However, training complex ML models on end-devices is impractical due to their constrained computing and energy capacities [5]. For example, a large CNN may take several minutes to process one single image on a mobile device [35]. Techniques like model compression [6] and model pruning [35] have been proposed to build compact ML models that can be deployed on lightweight end-devices. However, it is difficult and often impossible to obtain a high inference accuracy by training an ML model on an individual end-device due to the low quantity and/or diversity of training data [42].

Following a similar idea as data parallelism, federated learning [28] offers an architectural solution to training ML models on data distributed across multiple participants like mobile devices and Web-of-Things (WoT) devices [39]. It can be implemented under a centralized or a decentralized setting. The former requires a central parameter server that coordinates the training process and thus is subject to performance bottlenecks and single point failures [14]. Under the decentralized setting, participants must coordinate themselves to train a common ML model [17]. However, the issue of high resource and energy consumption remains [15].

Edge AI. Edge computing is a critical part of the 5G technology stack. It offers a fundamental solution to many challenges faced by cloud computing [38]. In an edge computing environment, edge servers are attached to base stations or access points within close proximity to end-devices like mobile devices [57]. End-devices can upload data to nearby edge servers for processing. This allows ML models to be deployed and trained on edge servers, which alleviates the resource and energy consumption incurred by ML model training on end-devices [44]. End-devices can also access services and retrieve data from nearby edge servers with low latency. Thus, ML inference can be performed on edge servers to inform real-time decision-making for end-devices and end-users in a variety of domains, e.g., social web [52], crowd sensing [61], food recognition [33] and web AR [34]. Taking this advantage, researchers are investigating ML model partitioning to strike a trade-off between inference delay and inference accuracy [18, 22, 58].

By processing data and provisioning services for nearby end-devices, edge servers offer context awareness [19]. This is a unique advantage that enables hierarchical ML applications for smart cities like transportation management, weather analysis, energy consumption management. For example, in a city that consists of multiple regions, ML models can be trained to predict road traffic and freeway traffic, i.e., the example application adopted in this paper for the presentation of Pyramid. ML models can also be trained to predict road traffic and freeway traffic to make region-level air quality predictions and city-level weather forecasts. Compared with existing ML frameworks, **the unique advantages of Pyramid are: 1) it facilitates the deployment of neural networks across the edge and the cloud to make both homogeneous (road traffic vs. freeway traffic) and heterogeneous (air quality vs. weather) hierarchical predictions; 2) it allows local predictions like road traffic and air quality predictions to be made in real time.**

Traffic Prediction. Traffic prediction is one of the core functions required for smart transportation management and has been studied intensively in the last two decades. Traditional approaches employ linear regression models such as ARIMA [51] and ARIMA [45]. These statistical models cannot well capture the complex traffic correlations and lose the competition to ML models based on stacked Auto-Encoder [37] and GCRN [30]. In 2017, Yu et al. proposed STGCN [56], the first graph neural network (GNN) based model for traffic predictions and achieved convincing results. After that, the ability of GNN to model spatial-temporal traffic correlations fueled the development of many GNN models, e.g., ASTGCN [12], GWNET [54], AGCRN [1], STGNN [50], LSGCN [20], STRN [31] and HGCN [11]. Compared with STGCN, these models consider more factors, e.g., road conditions, weather conditions, etc. These models advanced the accuracy of road traffic predictions, one better than the other. However, when used to make freeway traffic predictions, they cannot systematically capture the three types of freeway traffic correlations described in Section 3.2.1.

Inspired by data parallelism and model parallelism, Pyramid employs multiple Local Learners to make local predictions on edge servers in the corresponding regions, and a Global Learner to make global predictions on a cloud server. It is capable of capturing both intra-level and inter-level spatial-temporal correlations accurately and rapidly, as demonstrated in Section 4.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed Pyramid, a novel framework that enables homogeneous and heterogeneous hierarchical machine learning (ML) inferences. It leverages the unique advantages offered by edge computing to make accurate local and global predictions by jointly capturing intra-level spatial-temporal correlations and inter-level spatial-temporal correlations. It can also facilitate low-latency local inference. Two Pyramid NNs, one for road and freeway traffic predictions, and the other for air quality and weather forecasts were built. Extensive experiments were conducted to evaluate the performance of Pyramid on widely-used datasets demonstrate their superior performance against state-of-the-arts. In the future, we plan to capture other factors that have been proven relevant to road traffic, e.g., road conditions and weather, to investigate their impacts on freeway traffic prediction and the extensibility of Pyramid.

ACKNOWLEDGMENTS

This research is partially funded by Australian Research Council Discovery Projects (DP180100212 and DP200102491), and the National Science Foundation of China under Grants U20A20173 and 62125206. Qiang and Zeqian contributed equally to this research. Feifei Chen is the corresponding author of this paper.

REFERENCES

[1] L Bai, L Yao, C Li, X Wang, and C Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. In *34th Conference on Neural Information Processing Systems*.

[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *19th International Conference on Computational Statistics*. 177–186.

[3] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.

[4] Jiasi Chen and Xukan Ran. 2019. Deep learning with edge computing: a review. *Proc. IEEE* 107, 8 (2019), 1655–1674.

[5] Min Chen, Haichuan Wang, Zeyu Meng, Hongli Xu, Yang Xu, Jianchun Liu, and He Huang. 2020. Joint data collection and resource allocation for distributed machine learning at the edge. *IEEE Transactions on Mobile Computing* (2020). https://doi.org/10.1109/TMC.2020.3045436

[6] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. 2015. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*. PMLR, 2285–2294.

[7] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*. 1724–1734.

[8] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*. PMLR, 933–941.

[9] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Z Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, et al. 2012. Large scale distributed deep networks. In *25th International Conference on Neural Information Processing Systems*. 1223–1231.

[10] Sayda Elmi and Kian-Lee Tan. 2021. DeepFEC: energy consumption prediction under real-world driving conditions for smart cities. In *The Web Conference*. 1880–1890.

[11] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. 2021. Hierarchical graph convolution networks for traffic forecasting. In *36th AAAI Conference on Artificial Intelligence*, Vol. 35. 151–159.

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

[13] Jindong Han, Hao Liu, Hengshu Zhu, Hui Xiong, and Dejing Dou. 2021. Joint air quality and weather prediction based on multi-adversarial spatiotemporal networks. In *35th AAAI Conference on Artificial Intelligence*. 4081–4089.

[14] Rui Han, Shilin Li, Xiangwei Wang, Chi Harold Liu, Gaofeng Xin, and Lydia Y Chen. 2021. Accelerating Gossip-based Deep Learning in Heterogeneous Edge Computing Platforms. *IEEE Transactions on Parallel and Distributed Systems* 32, 7 (2021), 1591–1602.

[15] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[16] Qiang He, Guangming Cui, Xuyun Zhang, Feifei Chen, Shuiguang Deng, Hai Jin, Yanhui Li, and Yun Yang. 2019. A game-theoretical approach for user allocation in edge computing environment. *IEEE Transactions on Parallel and Distributed Systems* 31, 3 (2019), 515–529.

[17] 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.

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

[19] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computing—a key technology towards 5G. *ETSI white paper* 11, 11 (2015), 1–16.

[20] Rongzhou Huang, Chuyin Huang, Yubao Liu, Genan Dai, and Weiyang Kong. 2020. LSGCN: long short-term traffic prediction with graph convolutional networks.. In *International Joint Conference on Artificial Intelligence*. 2355–2361.

[21] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. PMLR, 448–456.

[22] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. 2017. Neurosurgeon: collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News* 45, 1 (2017), 615–629.

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

[24] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. 61––80.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[26] Seunghak Lee, Jin Kyu Kim, Xun Zheng, Qirong Ho, Garth A Gibson, and Eric P Xing. 2014. On model parallelization and scheduling strategies for distributed machine learning. In *27th International Conference on Neural Information Processing Systems*. 2834–2842.

[27] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation)*. 583–598.

[28] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[29] Youjie Li, Mingchao Yu, Songze Li, Salman Avestimehr, Nam Sung Kim, and Alexander Schwing. 2018. Pipe-SGD: a decentralized pipelined SGD framework for distributed deep net training. In *32nd International Conference on Neural Information Processing Systems*. 8056–8067.

[30] 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).

[31] Yuxuan Liang, Kun Ouyang, Junkai Sun, Yiwei Wang, Junbo Zhang, Yu Zheng, David Rosenblum, and Roger Zimmermann. 2021. Fine-grained urban flow prediction. In *The Web Conference*. 1833–1845.

[32] Yuhua Lin and Haiying Shen. 2017. CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. *IEEE Transactions on Parallel and Distributed Systems* 2 (2017), 431–445.

[33] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, Ma Yunsheng, Songqing Chen, and Peng Hou. 2017. A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. *IEEE Transactions on Services Computing* 11, 2 (2017), 249–261.

[34] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[35] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision*. 2736–2744.

[36] Zongqing Lu, Swati Rallapalli, Kevin S Chan, Shiliang Pu, and Tom La Porta. 2021. Augur: modeling the resource requirements of ConvNets on mobile devices. *IEEE Transactions on Mobile Computing* 107, 2 (2021), 352–365.

[37] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2014. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2014), 865–873.

[38] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. 2017. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2322–2358.

[39] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[40] Benedikt Ostermaier, Kay Römer, Friedemann Mattern, Michael Fahrmair, and Wolfgang Kellerer. 2010. A real-time search engine for the web of things. In *2010 Internet of Things (IOT)*. IEEE, 1–8.

[41] Zheyi Pan, Songyu Ke, Xiaodu Yang, Yuxuan Liang, Yong Yu, Junbo Zhang, and Yu Zheng. 2021. AutoSTG: neural architecture search for predictions of spatio-temporal graph. In *Web Conference*. 1846–1855.

[42] Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. 2019. Wireless network intelligence at the edge. *Proc. IEEE* 107, 11 (2019), 2204–2239.

[43] 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.

[44] Yuanming Shi, Kai Yang, Tao Jiang, Jun Zhang, and Khaled B Letaief. 2020. Communication-efficient edge AI: algorithms and systems. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2167–2191.

[45] Hoo-Chang Shin, Matthew R Orton, David J Collins, Simon J Doran, and Martin O Leach. 2012. Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2012), 1930–1943.

[46] Alex J Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and Computing* 14, 3 (2004), 199–222.
[47] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *27th International Conference on Neural Information Processing Systems*. 3104–3112.
[48] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2464–2469.
[49] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2017. Distributed deep neural networks over the cloud, the edge and end devices. In *37th IEEE International Conference on Distributed Computing Systems*. IEEE, 328–339.
[50] 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 *The Web Conference*. 1082–1092.
[51] Billy M Williams and Lester A Hoel. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering* 129, 6 (2003), 664–672.
[52] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. 2019. Machine learning at Facebook: understanding inference at the edge. In *IEEE International Symposium on High Performance Computer Architecture*. IEEE, 331–344.
[53] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4820–4828.
[54] Z Wu, S Pan, G Long, J Jiang, and C Zhang. 2019. Graph WaveNet for deep spatial-temporal graph modeling. In *28th International Joint Conference on Artificial Intelligence (IJCAI)*. 1907–1913.
[55] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: a meta-learning approach for spatial-temporal prediction. In *The Web Conference*. 2181–2191.
[56] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *27th International Joint Conference on Artificial Intelligence*. 3634–3640.
[57] Liang Yuan, Qiang He, Siyu Tan, Bo Li, Jiangshan Yu, Feifei Chen, Hai Jin, and Yun Yang. 2021. CoopEdge: A Decentralized Blockchain-based Platform for Cooperative Edge Computing. In *The Web Conference*. 2245–2257. https://doi.org/10.1145/3442381.3449994
[58] Letian Zhang, Lixing Chen, and Jie Xu. 2021. Autodidactic Neurosurgeon: collaborative deep inference for mobile edge intelligence via online learning. In *The Web Conference*. 3111–3123.
[59] Xiyue Zhang, Chao Huang, Yong Xu, Lianghao Xia, Peng Dai, Liefeng Bo, Junbo Zhang, and Yu Zheng. 2020. Traffic flow forecasting with spatial-temporal graph diffusion network. In *35th AAAI Conference on Artificial Intelligence*. 15008–15015.
[60] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: an extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on computer Vision and Pattern Recognition*. 6848–6856.
[61] Zhenyu Zhou, Haijun Liao, Bo Gu, Kazi Mohammed Saidul Huq, Shahid Mumtaz, and Jonathan Rodriguez. 2018. Robust mobile crowd sensing: when deep learning meets edge computing. *IEEE Network* 32, 4 (2018), 54–60.

A WEATHER ANALYSIS

To demonstrate that Pyramid is also applicable to hierarchical inferences in other domains in addition to traffic prediction, we build, train and test a Pyramid NN named Pyramid-WA to make air quality predictions at the edge and weather forecasts in the cloud.

A.1 Experiment Setup

Dataset. Pyramid-WA is trained on the public KDD18-Beijing dataset[1]. This dataset contains hourly air quality information measured by 5 significant pollutants over 13 months between 2017-01-01 and 2018-01-31, including $PM_{2.5}$, PM_{10}, NO_2, CO and O_3. The data is collected from 35 air quality monitoring stations deployed across five regions in Beijing city. It also contains the weather information of 10 classes, e.g., Sunny, Rainy, Cloudy, etc., over the same 13 months

Environment. The experiments on Pyramid-WA are conducted in the same environment described in Section 4.1, including five edge servers and a cloud server.

Baselines. Pyramid-WA is evaluated against LSTM [47], GRU [7], DDNN [49] and HGCN [11]. ARIMA [51] and SVR [46] are not included because they are not designed to perform classifications.

Experiment Settings. The dataset is split into three parts, 70% of the data for training, 20% for test and 10% for validation. Similar to the settings described in Section 4.1, LSTM, GRU and DDNN models are trained on air quality data to predict air quality, on weather data to forecast weather. HGCN is trained in a different way from the experiments on Pyramid-TP. Two HGCN models are trained, one on air quality data to predict air quality and the other on weather data to forecast weather. All the approaches predict the air quality and the weather hourly.

When testing Pyramid-WA, similar to Pyramid-TP, Local Learners are trained on edge servers to predict air quality at the 35 air quality monitoring stations. Global Learner is trained on the cloud server to forecast weather in the five regions. In general, weather impacts air quality, but not the other way around. For example, on rainy days, some air pollutants and pollen in the air are washed away, which increases the air quality. To capture such inter-level correlations, weather features extracted by Global Learner are fed to Local Learners for training, in addition to historical air quality data.

Performance Metrics. Air quality prediction is a regression prediction, similar to traffic prediction. Thus, MSE, MAE, RMSE and R-Squared are employed to evaluate the accuracy of Local Learners. Weather forecast is classification. Thus, MLE and precision are employed to evaluate the accuracy of Global Learner.

[1]https://biendata.com/competition/kdd_2018/data/

A.2 Experiment Results

Table 3 presents the results. We can see that Pyramid-WA achieves the highest prediction accuracy in almost all the cases, outperforming the other approaches significantly. Three key conclusions can be drawn. 1) Pyramid can support hierarchical inferences in various domains, capturing both intra-level and inter-level temporal-spatial correlations accurately and rapidly. 2) Pyramid can facilitate homogeneous and heterogeneous hierarchical inferences. Pyramid-TP makes road traffic predictions and freeway traffic predictions, which are homogeneous. Pyramid-WA makes air quality predictions and weather forecasts, which are heterogeneous. 3) Pyramid can capture inter-level spatial-temporal correlations both ways. Pyramid-TP includes road traffic features in freeway traffic predictions. Pyramid-WA includes weather features in air quality predictions.

The comparison between Pyramid-TP and the baselines in model convergence, training time and inference time is similar to what was presented in Section 4.2. It is thus omitted here.

Table 3: Prediction Accuracy

	Local Prediction (Air Quality Index)								Global Prediction (Weather)			
	MSE	MAE	RMSE	R-Squared	MSE	MAE	RMSE	R-Saured	MLE	Precision	MLE	Precision
	500 Epochs				1,000 Epochs				500 Epochs		1,000 Epochs	
LSTM [47]	0.33	3.66	5.80	33.33%	0.25	3.84	5.05	49.51%	0.90	43.21%	0.91	43.21%
GRU [7]	0.11	2.18	3.85	53.16%	0.09	2.01	2.85	85.99%	0.63	68.95%	0.32	78.57%
DDNN [49]	0.36	2.13	3.15	56.85%	0.22	2.08	3.77	67.88%	2.25	43.21%	1.70	43.21%
HGCN [11]	0.10	2.13	3.16	59.13%	0.09	2.26	2.83	85.11%	0.91	43.21%	0.22	68.80%
Pyramid-WA	0.09	2.01	2.95	67.39%	0.09	2.09	2.83	90.78%	1.51	79.28%	1.40	80.92%