Mixture of Experts based Model Integration for Traffic State Prediction

Rajarshi Chattopadhyay and Chen-Khong Tham
Department of Electrical and Computer Engineering
National University of Singapore
E-mail: ceerc@nus.edu.sg, eletck@nus.edu.sg

Abstract—**Traffic forecasting is an important part of many future intelligent transportation systems and can be particularly useful for planning and navigation applications. The task is challenging because of the complex spatial patterns of road networks and dynamic temporal nature of traffic conditions. Recently, deep learning architectures with specially designed graph convolution layers to extract spatial patterns and recurrent or temporal convolution layers to extract temporal patterns have achieved good results for this task. In this paper, we propose a Mixture of Experts (MoE) based model integration framework to enhance the performance of these state-of-the-art traffic prediction models. In addition, we propose a novel entropy based loss function to improve the training of the MoE ensemble. Our experiments show that the performance of the Spatio-Temporal Graph Convolution Network (STGCN), a state of the art model, can be significantly improved.**

Index Terms—**Intelligent Transportation; Vehicular and Transportation Data Analytics; Traffic Forecasting; Graph Convolutional Networks; Mixture of Experts**

I. Introduction

A lot of current research has been focused on Intelligent Transportation Systems (ITS). Smart navigation applications have the potential to reduce the end-to-end travel time of vehicles. Traffic state prediction on road networks is a crucial component of these applications. It can help traffic authorities to plan custom signalling schemes (considering future traffic states) to control traffic flows and reduce congestion [1]. In addition, it can reduce the commute time of individual vehicles as navigation applications can now plan time optimal routes based on both the current and future predicted traffic states [2]. A more accurate traffic state prediction algorithm is clearly beneficial for both these scenarios.

Typically, the traffic state of a road network is characterised by measuring the fundamental variables of traffic flow, e.g. average speed, volume, and density at each road segment at any time. Furthermore, based on the prediction interval, the traffic state prediction problem is generally classified into two categories: short-term (5 to 30 minutes) and long term (over 30 minutes). While traditional statistical and time series methods like linear regression, Support Vector Machines (SVM) and Autoregressive Integrated Moving Average (ARIMA) perform well for short-term prediction, they are unable to capture the complex long term spatio-temporal patterns in traffic data.

Recently the use of Deep Learning (DL) has become more popular in this domain. The specific DL methods being applied keep evolving with the advancements made in DL in other areas. One of the first works applying DL to traffic state prediction is described in [3]. The authors use a Stacked Autoencoder (SAE) followed by a logistic regression layer. It was shown to outperform traditional methods like SVM, random-walk forecast and shallow neural networks. After this initial success more specialized models like Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM) networks were applied. It was well known from their usage in other application domains that CNN and LSTMs are good for extracting spatial and temporal patterns from data, respectively. The work in [4] concatenates features extracted by CNN and LSTM layers and feeds them into a linear regression layer to make the final predictions. The proposed model was shown to outperform the previous state-of-the-art, viz. SAE.

The standard CNN is designed for euclidean data, e.g. images. However, historical traffic flow data on road networks is more of a graph signal. Spatial [5] and spectral graph convolution networks [6], [7] were especially designed to process graph signals. The use of graph convolution resulted in the development of new kinds of models where spatial features are extracted by graph convolution and these features are then fed into the recurrent layers for temporal feature extraction. Diffusion Convolution Recurrent Neural Network (DCRNN) [8] was one of the first works applying graph convolution to traffic data. It treated the the graph convolution as a diffusion process and further considers the road network as a directed graph. Spatio-Temporal Graph Convolution Network (STGCN) [9] further replaces the recurrent layer with temporal convolutions to take full advantage of parallel hardware to train deep networks. Both DCRNN and STGCN were shown to have superior performance compared to previously considered model architectures. In the meantime, the concept of self-attention was becoming widely used in the area of natural language processing. The work in [10], extends the STGCN model by adding a global attention layer. This model is referred to as ASTGCN. It was shown to outperform STGCN. More recently, self-attention based transformer neural network [11] has been applied to traffic forecasting. The main idea is to replace the graph convolution layer with several multi-headed attention layers. The idea of self-attention had been previously applied to graphs using Graph Attention Networks (GAT) [12]. However, GAT only uses self-attention to compute the edge weights between neighbouring nodes. It still retains the basic idea of graph convolution. The self-attention based spatio-temporal transformer network [13] was shown to perform slightly better than the STGCN model.

Ever since its introduction, the Mixture of Experts (MoE) [14] model integration framework has been applied to many classification and regression problems. It consistently gives improved performance compared to individual models. However, to the best of our knowledge, application of the MoE framework has not been explored in the area of traffic forecasting. In this paper, we propose a MoE based model integration framework to enhance the performance of the existing state-of-the-art models. We also propose a novel loss function for training these ensembles. We attained significant performance improvement in our experiments. The rest of the paper is organized as follows. In Section II we define the problem and describe our solution methodology. Section III describes our experiments and a discussion of the results. Finally, Section IV concludes the paper and mentions future research directions.

II. Methodology

A. Road Network as a Graph

The road network can be defined as an undirected graph $G = (V, E, A)$, where V is a finite set of $|V| = N$ nodes; E is a set of edges, indicating the connectivity between the nodes; $A \in R^{N\times N}$ denotes the adjacency matrix of graph G . Each node on the traffic network G detects F measurements with the same sampling frequency, that is, each node generates a feature vector of length F at each time step.

B. Traffic State Prediction on Road Networks

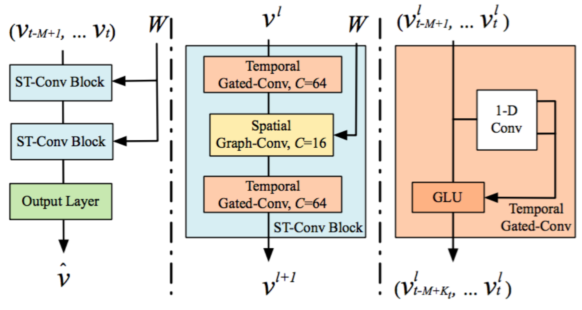
The goal of traffic forecasting is to use the previously observed traffic state to forecast a fixed length of future traffic state in a traffic network. In this work, the traffic state is the average speed on road segments. Suppose that the entire traffic network under consideration has N road segments. Then, the traffic state at a particular time step can be denoted by a N-dimensional vector v_t, whose elements consists of the average speed on the individual road segments. If the average speed is recorded over the past T time steps, including the current step, then we get a sequence of vectors v_{t-T+1}, v_{t-T+2}, .., v_t. The goal is to predict the most likely length P vector sequence in the future given the past T observations (1).

$$\begin{aligned} &v_{t+1}, v_{t+2}, .., v_{t+P} = \\ &\quad \underset{v_{t+1}, v_{t+2}, .., v_{t+P}}{\arg\max}\; P(v_{t+1}, v_{t+2}, .., v_{t+P} | v_{t-T+1}, v_{t-T+2}, .., v_t) \end{aligned} \tag{1}$$

C. Spatio-Temporal Graph Convolution Network

The STGCN model uses spectral graph convolution and 1-D causal convolutions to model the spatial and temporal dependencies in traffic data, respectively.

1) Spectral Graph Convolution: The standard convolution kernels are well suited to be applied to regular euclidean grids like images. However, they cannot be directly applied to graph structured data like road networks. The idea of spectral graph convolutions was first introduced in [6]. The authors suggested the use of graph Fourier transform to capture the spatial dependencies in data. The spectral graph convolution of a graph signal x with kernel Φ is defined as in (2). Here, U is



$(v_{t-M+1}, \dots v_t)$
W
ST-Conv Block
ST-Conv Block
Output Layer
$\hat{v}$
v^l
W
Temporal Gated-Conv, C=64
Spatial Graph-Conv, C=16
Temporal Gated-Conv, C=64
ST-Conv Block
v^{l+1}
$(v^l_{t-M+1}, \dots v^l_t)$
1-D Conv
GLU
Temporal Gated-Conv
$(v^l_{t-M+K_t}, \dots v^l_t)$

Fig. 1. STGCN model structure with speed inputs [9].

the matrix of eigenvectors of the normalized graph Laplacian L and Λ is the diagonal matrix of its eigenvalues.

$$\Phi *_G x = \Phi(L)x = U\Phi(\Lambda)U^T x \tag{2}$$

The calculation of U is computationally expensive. Therefore, a Chebyshev polynomial based approximation was proposed [7]. Using this approximation, spectral graph convolution can be approximated as in (3). Here, θ_k are free parameters, $\tilde{L}$ is the graph Laplacian scaled w.r.t. the maximum eigenvalue and $T_k(\tilde{L})$ are Chebyshev polynomials of order k and Q denotes the order of approximation.

$$\Phi *_G x \approx \sum_{k=0}^{Q-1} \theta_k T_k(\tilde{L})x \tag{3}$$

The graph convolution approximation can be generalised to multi-channel inputs and outputs as in (4). Here, $\Theta_{i,j} \in R^Q$ are free parameters and C_i and C_o are the number of channels in the input and output, respectively.

$$y_j = \sum_{i=1}^{C_i} \Theta_{i,j}(L)x_i, 1 \le j \le C_o \tag{4}$$

2) Temporal Convolutions: The temporal convolution kernel used in STGCN are 1-D causal convolutions filters of width L_t. Temporal convolution with kernel ρ can be mathematically represented as in (5). Here Y denotes the input to the temporal convolution operation. The output dimension is $(T - L_t + 1) \times C_o$.

$$\Gamma *_\rho Y = \rho * Y \tag{5}$$

3) Spatial-temporal Convolution Block: The convolution operations defined in (4) and (5) are used in the form of specialised blocks within STGCN. Each block contains a spatial convolution operation sandwiched between two temporal convolution blocks as shown in Figure 1. Mathematically, a spatio-temporal convolution block can be represented as in (6). Here, v^l denotes the output of layer l and ReLU() denotes the rectified linear operation.

$$v^{l+1} = \Gamma_1^l *_\tau \mathrm{ReLU}(\Theta^l *_G (\Gamma_0^l *_\tau v^l)) \tag{6}$$

4) Output Layer: The last layer of the STGCN architecture consists of a temporal convolution kernel as defined in (5) and is followed by a fully connected layer. The temporal convolution flattens the features to a single time step and then the fully connected layer combines the multi-channel information (if any) into a single dimension.

Figure 1 shows the overall architecture of the STGCN model. It consists of two spatio-temporal convolution blocks followed by an output layer.

D. Mixture of Experts Framework

1) Model Structure: MoE is an ensemble learning approach, initially developed in the field of neural networks. It was established based on the divide-and-conquer principle in which the problem space is divided between several neural network experts, supervised by a gating network. Figure 2 shows the general MoE model architecture. In our context, X denotes the input average speed matrix for the road network with the number of rows equal to the lookback window size and the number of columns equal to the number of road segments. It can be written as in (7). The individual experts are prediction models and each one makes a prediction for a given input. We use y_i to denote the output of expert i. In addition, the gating network outputs a set of values for a given input, one for each expert, i.e. if there are 3 experts, the gating networks output layer will have 3 nodes. These are denoted by g_i. The overall output of the MoE model, denoted by Y, is computed as in (8). Here, K denotes the number of experts.

$$X = [v_{t-T+1}, v_{t-T+2}, .., v_t]^T \tag{7}$$

$$Y = \sum_{i=1}^{K} g_i . y_i \tag{8}$$

The exact model structure will depend upon the choice of experts and the gating network. Our objective in this work is to improve the performance of STGCN by constructing ensemble models using the MoE framework. Our gating network is a three layer deep neural network with 512 nodes in each hidden layer. Henceforth, we will refer to the MoE based STGCN ensemble as the "MoE model". More specifically, we want to study the performance of the MoE model for different number of MoE experts and also design suitable loss functions for their effective training.

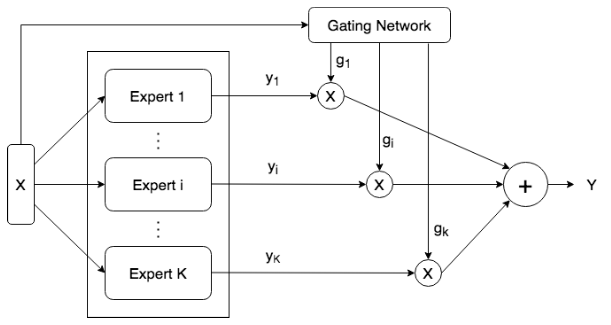


Fig. 2. Model Integration using the Mixture of Experts paradigm.

2) Loss Functions: One option is to minimize the squared error loss between the target values and Y as in (9).

$$E = \frac{1}{2} \sum_{j=1}^{|TS|} |T^j - Y^j|^2 \tag{9}$$

Here, T^j and Y^j are the ground-truth and prediction values, respectively, corresponding to training sample j. $|TS|$ is the total number of training samples. If we use this error function, the weights of each expert are updated based on the overall ensemble error, rather than the errors of each expert. This strong coupling in the process of updating the weights of the experts encourages cooperation among the experts over the whole problem space and will cause almost all of the experts to be employed for each data sample. This would not be consistent with the desired localisation of the experts in different regions of the input space.

Another option is to use the loss function stated in (10).

$$E = \frac{1}{2} \sum_{j=1}^{|TS|} \sum_{i=1}^{K} g_i^j . |T_i^j - y_i^j|^2 \tag{10}$$

Here, y_i^j is the prediction of the i^{th} expert corresponding to training sample j. Furthermore, g_i^j is the output of the gating network corresponding to expert i and training sample j. This loss function leads the different experts to specialize in different regions of the input space. The regions are decided by the gating network as the weights of a particular expert is updated only when g_i^j is high. We have chosen to use this loss function.

3) Adding Entropy to the Loss Function: While training the MoE model using the loss function in (10), we observed that one expert tends to dominate for all the training samples. The output layer of the gating network is typically a softmax layer. The gating network output corresponding to the dominating expert was observed to be close to 1 and for the other experts it was close to 0. Later, we will describe this behavior in detail when presenting our experimental results. The main motivation behind using MoE based ensemble framework is to have different experts specialize in different regions of the input space. This is not achieved if one expert dominates the prediction outcome for all training samples. One way to ensure that a single expert does not dominate is by adding the negative entropy of the gating network output distribution to the loss function in (10). The entropy of a distribution is the measure of its randomness. Enforcing a higher entropy on the gating network output distribution causes the gating network output to be less skewed. The entropy of the output distribution for a particular training sample j can be written as in (11). Here, GN denotes the gating network.

$$H(GN, j) = -\sum_{i=1}^{K} g_i^j . \log(g_i^j) \tag{11}$$

Therefore, the total entropy for all the training samples can be computed as in (12).

$$H = \sum_{j=1}^{|TS|} H(GN, j) \tag{12}$$

The overall loss function after adding the total entropy can

TABLE I
COMPARISON OF PER EPOCH TRAINING TIME (SECONDS) ON GPU PLATFORM VS IPU PLATFORM

Model	GPU platform	IPU platform	Number of Parameters
DNN	1.1869	**0.2712**	414,308
CNN	1.1830	**0.2764**	464,056
LSTM	1.2613	**0.3837**	415,588
STGCN	5.1001	**1.6223**	393,220
MoE (2 STGCN Experts)	9.1480	**2.6174**	2,725,066
MoE (3 STGCN Experts)	13.1744	**3.6623**	3,118,799
MoE (4 STGCN Experts)	17.2285	**6.2179**	3,512,532

TABLE II
PERFORMANCE COMPARISON FOR THE PEMSD7(M) DATASET

Model	MAE (15/30/45 min)	MAPE (%) (15/30/45 min)	RMSE (15/30/45 min)
DNN	3.597/3.771/3.961	9.283/9.851/10.516	6.084/6.321/6.629
CNN	3.591/4.087/4.698	9.038/10.138/11.655	5.962/6.607/7.536
LSTM	3.534/3.660/3.854	8.846/9.101/9.602	6.003/6.203/6.556
STGCN	2.256/3.037/3.578	5.263/7.334/8.692	4.049/5.714/6.778
MoE (2 STGCN Experts)	2.036/2.756/3.316	4.590/6.504/8.018	3.811/5.398/6.526
MoE (3 STGCN Experts)	2.018/**2.713**/3.266	**4.544**/**6.451**/**7.850**	**3.788**/**5.332**/**6.466**
MoE (4 STGCN Experts)	**2.016**/2.715/**3.251**	4.559/6.461/7.959	3.804/5.372/6.467

TABLE III
STANDARD DEVIATION OF RESULTS FOR THE PEMSD7(M) DATASET (10 EXPERIMENTS)

Model	MAE (15/30/45 min)	MAPE (%) (15/30/45 min)	RMSE (15/30/45 min)
STGCN	0.0001/0.0018/0.0115	0.0012/0.0651/0.0075	0.0002/0.0505/0.0159
MoE (2 STGCN Experts)	0.0003/0.0020/0.0068	0.0005/0.0034/0.0177	0.0001/0.0032/0.0157
MoE (3 STGCN Experts)	0.0001/0.0009/0.0049	0.0010/0.0265/0.0068	0.0006/0.0020/0.0059
MoE (4 STGCN Experts)	0.0001/0.0006/0.0024	0.0019/0.0085/0.0214	0.0003/0.0019/0.0085

be written as (13). Here, α is a hyperparameter that determines the weight of the entropy term.

$$E_{en} = E - \alpha.H \tag{13}$$

While adding $-\alpha.H$ to the loss function discourages the dominance of one expert over the entire input space, it also discourages different experts from specializing in different regions of the input space. Assume that we are training an MoE model with two experts. Ideally, we should have expert 1 specialize on some input subspace R_1 and expert 2 on input subspace R_2. More precisely, this is equivalent to $g_1^j \approx 1, g_2^j \approx 0 \; \forall j \in R_1$ and $g_1^j \approx 0, g_2^j \approx 1 \; \forall j \in R_2$. However, note that maximizing the entropy term in (12) works against attaining this objective as in both cases we will have $H(GN, j) \approx 0$. We can also write $H = \sum_{j \in R_1} H(GN, j) + \sum_{j \in R_2} H(GN, j)$. This implies $H \approx 0$. It is easy to see that H is maximized when $g_i^j \approx \frac{1}{K} \; \forall i, j$. To address these issues, we modify the entropy term as shown in (14):

$$H = -\sum_{i=1}^{K} \Big(\frac{\sum_{j=1}^{|TS|} g_i^j}{|TS|} \Big) . \log \Big(\frac{\sum_{j=1}^{|TS|} g_i^j}{|TS|} \Big) \tag{14}$$

To maximize this expression, a particular expert will dominate over some subspace of the input space while being dormant in others, provided each expert has dominance over some subspace, which is the desired behavior.

III. EXPERIMENTS

A. Dataset

In our experiments, we used the PeMSD7(M) dataset [9] to validate the performance of the MoE model. PeMSD7 was collected from the Caltrans Performance Measurement System (PeMS) by over 39,000 sensor stations, deployed across the major metropolitan areas of the California state highway system and contains weekday traffic data. The PeMSD7(M) dataset is further selected from PeMSD7 by randomly selecting a smaller district with 228 road segments.

B. Data Preprocessing

The standard time interval in the PeMSD7(M) dataset is set to 5 minutes. Thus, every node of the road graph contains 288 data points per day. The linear interpolation method is used to fill missing values. The entire dataset is then split into training, validation and test sets with the split ratio set to $0.7 : 0.15 : 0.15$. In addition, the data is z-score normalized based on the mean and variance of the training set. The STGCN model needs the adjacency matrix of the road network. The adjacency matrix of the road graph is computed based on the distances between stations in the traffic network. The weighted adjacency matrix W can be computed as in (15).

$$w_{ij} = \begin{cases} \exp(-\frac{d_{ij}^2}{\sigma^2}) & ; i \neq j \text{ and } \exp(-\frac{d_{ij}^2}{\sigma^2}) \geq \varepsilon \\ 0 & ; \text{otherwise} \end{cases} \quad (15)$$

Here d_{ij} is the physical distance between stations i and j. σ^2 and ε are thresholds to control the distribution and sparsity of matrix W, and are assigned values 10 and 0.5, respectively.

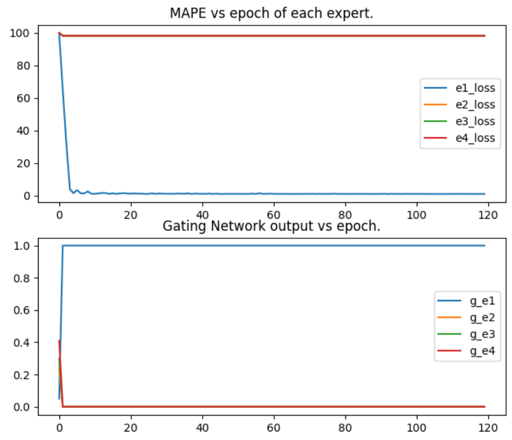


Fig. 3. Performance of experts for random sample 1 (baseline loss function).

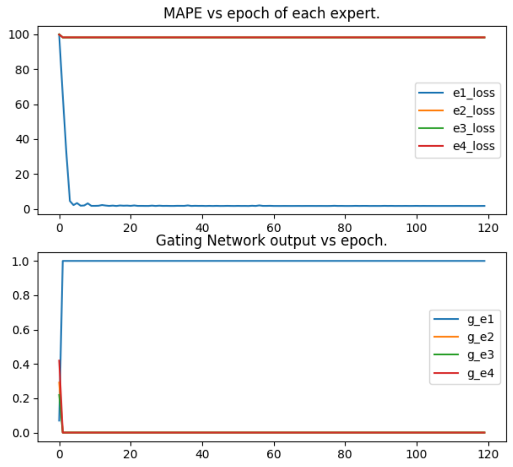


Fig. 4. Performance of experts for random sample 2 (baseline loss function).

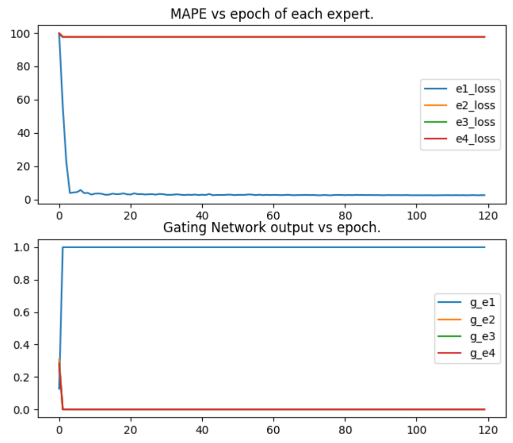


Fig. 5. Performance of experts for random sample 3 (baseline loss function).

C. Experimental Settings

The lookback window size was set to 60 minutes, i.e. 12 observed data points ($T = 12$) are used to forecast traffic conditions in the next 15, 30, and 45 minutes ($P = 3,6,9$). We have used the Mean Absolute Error (MAE), Mean Absolute Percentage Errors (MAPE), and Root Mean Squared Error (RMSE) metrics to compare and evaluate the performance of the MoE model. The RMSProp optimizer was used for training the models. We evaluated the performance of a single STGCN model vs using $2,3$ and 4 STGCN experts in the MoE model. In addition, we evaluated the performance of deep neural network (DNN), CNN and LSTM models.

D. Implementation Details and Training Procedure

The single STGCN model was implemented using the code provided in Github by the authors of the original STGCN paper [9]. In order to implement the MoE model, we modified the original STGCN code in significant ways. Firstly, we rewrote the code in Keras. The spatial and temporal convolution layers of STGCN were re-written by sub-classing the Keras Layer class. This allowed us to define STGCN as a Keras model. Written as a Keras model, it becomes convenient to add a gating network and combine several STGCN models in the MoE framework. In addition, some changes were required to ensure compatibility with Tensorflow 2.5.0.

Secondly, we developed custom Keras callbacks to analyse the performance of each individual expert and the gating network during training. This enabled us to observe the dominance of certain experts and helped motivate the entropy enhanced loss function proposed in (13). We also implemented this as a Keras custom loss function. Last but not least, several cascaded layers were defined in the original STGCN code, e.g. the spatio-temporal convolution block and the output layer. Cascaded layers help in simplifying the code. However, they have compatibility problems with the model save functionality

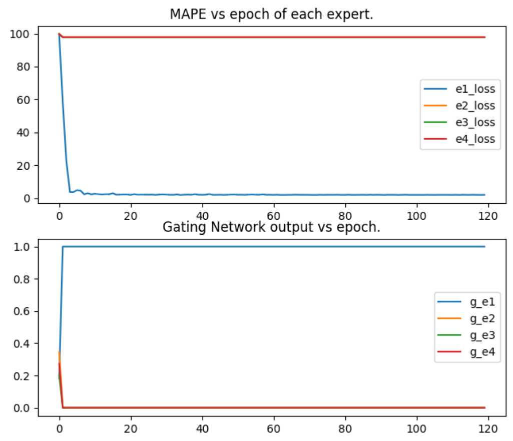


Fig. 6. Performance of experts for random sample 4 (baseline loss function).

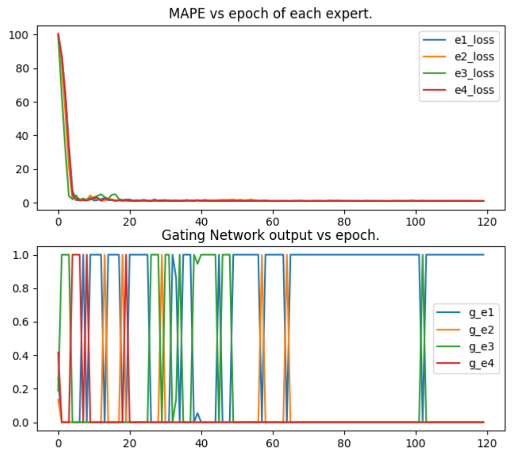


Fig. 7. Performance of experts for random sample 1 (entropy loss function).

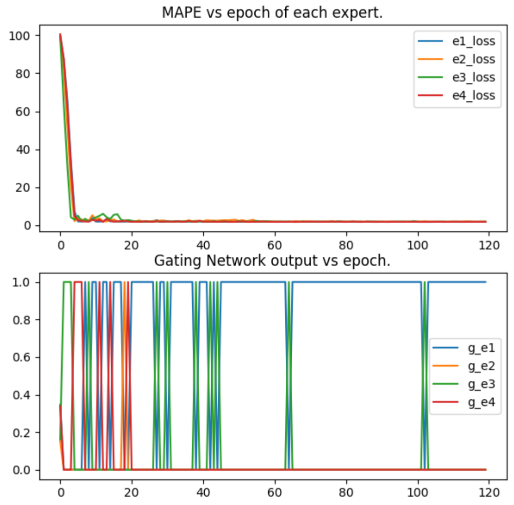


Fig. 8. Performance of experts for random sample 2 (entropy loss function).

of Keras and we made the required changes to resolve this issue.

The DNN, CNN, LSTM and MoE models were trained for 120 epochs with a batch size of 50. The loss function in (13) with modified H (14) was used to train the MoE models while the other models were trained using the standard mean squared error loss. The initial learning rate was set to 0.001 and staircase decay was used with a step-size of 20 epochs and a decay rate of 0.6.

E. Hardware Platforms

We trained and executed the models described above on two hardware platforms. Firstly, we trained our models on Google Colab with Graphics Processing Unit (GPU) hardware acceleration (Intel Xeon CPU @ 2.20 GHz and NVIDIA Tesla V100 SXM2 16 GB GPU). In addition, we trained our models on Graphcore Intelligence Processing Units (IPUs) [15] on IPU-POD16 server blades on Graphcloud. The IPU-POD16 server blades have 16 GC200 IPU processors, and the host CPU server has two AMD EPYC 7742 CPUs with 512 GB of RAM.

The per epoch training time comparison is shown in Table I. In our experiments, the IPU-based platform was 3 to 4 times faster than the GPU-based platform for the models shown.

F. Results

The results are summarized in Table II. It can be seen that for the PeMSD7(M) dataset, the MoE models outperforms all the individual models. The MoE model with 3 STGCN experts gives the best performance in most cases. We trained each model 10 times and Table II and Table III show the average results and standard deviation, respectively. The low values of standard deviation indicates the stability of the training method.

Next, we examine the benefits of using the entropy enhanced error function when training our model. We selected four training samples randomly from the training set and observed the gating network outputs and the squared error losses of the individual experts for the MoE model with 4 STGCN experts. The plots in Figures 3 to 6 show the results when the MoE model was trained using the loss function in (10). Here *ek_loss* and *g_ek* denote the value of the loss function using the MAPE metric and gating network output corresponding to expert k and the particular training sample. We see in Figures 3 to 6 that the gating network output corresponding to expert 1 always dominates, i.e. it is close to 1, whereas the gating network output corresponding to all the other experts is close to 0. The corresponding values of the individual loss functions also show that experts 2 to 4 are not trained and the gating network assigns all the weight to expert 1. The overall performance of this model is close to that of the individual STGCN model.

The plots in Figures 7 to 10 show the results when the MoE model with four experts was trained using the entropy

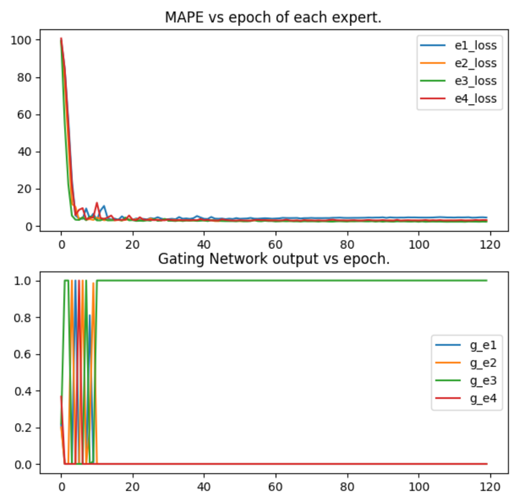


Fig. 9. Performance of experts for random sample 3 (entropy loss function).

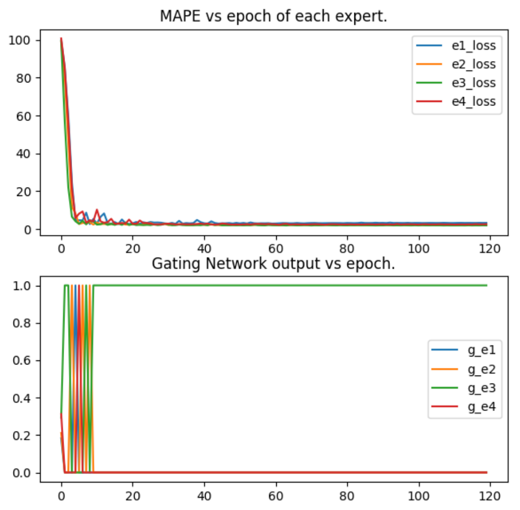


Fig. 10. Performance of experts for random sample 4 (entropy loss function).

enhanced loss function in (13) with H defined in (14). It can be seen that one expert no longer dominates, and unlike before, all the experts have the opportunity to be trained and take responsibility for different training samples. Notice in Figures 9 and 10 that expert 3 is the dominating expert and the individual losses of experts 1, 2 and 4 are slightly higher than expert 3. This shows that while experts 1, 2 and 4 have been trained, they do not specialize for this training sample, whereas expert 3 does. The specialization of different experts for different training samples is what gives the MoE model their superior performance. It can be seen that all the individual experts have fairly close performance in Figures 7 and 8. Therefore, the gating network chooses different experts at different times for these training samples and its output fluctuates. Lastly, when the number of experts is increased, each expert effectively gets a smaller number of training samples and the overall prediction accuracy declines.

IV. Conclusion and Future Work

In this paper, a MoE based ensemble modelling method is proposed to enhance the performance of the STGCN traffic state prediction model. We overcame the difficulty of training an MoE based STGCN by proposing a novel entropy based loss function. Our experiments show a significant improvement in performance over the standalone STGCN. In future work, we plan to verify the effectiveness of the proposed framework using other datasets and consider other state-of-the-art models, as well as explore other structures for the gating network used in the MoE model. In addition, we plan to validate the inverse relationship between future traffic state prediction accuracy and the end-to-end travel time of vehicles.

V. Acknowledgments

The authors acknowledge Graphcore for providing technical advice and IPU computational resources on Graphcloud.

References

[1] A. Ahmed, S. A. A. Naqvi, D. Watling, and D. Ngoduy, “Real-time dynamic traffic control based on traffic-state estimation,” *Transportation Research Record*, vol. 2673, no. 5, pp. 584–595, 2019.
[2] T. Liebig, N. Piatkowski, C. Bockermann, and K. Morik, “Dynamic route planning with real-time traffic predictions,” *Information Systems*, vol. 64, pp. 258–265, 2017.
[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, “Traffic flow prediction with big data: a deep learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2014.
[4] Y. Wu and H. Tan, “Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework,” *arXiv:1612.01022*, 2016.
[5] M. Edwards and X. Xie, “Graph based convolutional neural network,” *arXiv:1609.08965*, 2016.
[6] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, “Spectral networks and locally connected networks on graphs,” in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.
[7] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” *Advances in Neural Information Processing Systems*, vol. 29, pp. 3844–3852, 2016.
[8] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *International Conference on Learning Representations (ICLR ’18)*, 2018.
[9] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.
[10] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatial-temporal graph convolutional networks for traffic flow forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.
[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
[12] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” *International Conference on Learning Representations*, 2018, accepted as poster. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ
[13] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, “Spatial-temporal transformer networks for traffic flow forecasting,” *arXiv:2001.02908*, 2020.
[14] S. E. Yuksel, J. N. Wilson, and P. D. Gader, “Twenty years of mixture of experts,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1177–1193, 2012.
[15] “Graphcore Intelligence Processing Unit (IPU).” [Online]. Available: https://www.graphcore.ai/