Deep Learning for Network Traffic Prediction

Wenxin Fang - wxfang@ucdavis.edu
Cheng-Hao Tsai - chhtsai@ucdavis.edu
Yijia Wang - yijwa@ucdavis.edu
Electrical and Computer Engineering
University of California, Davis, USA

June 10, 2024

Abstract

This project implemented three different deep-learning approaches (LSTM, GNN, Transformer) to analyze the real-life data from the city of Milan. We then compared the fitting loss, running time, and testing accuracy to discuss which method is the best solving network traffic prediction problems.

1 Background and Literature

With the emergence of edge computing, network traffic is recently becoming increasingly complex and diversified, presenting new challenges for network traffic prediction. Network traffic prediction such as bandwidth utilization can assist service providers in predicting future network congestion so that they can take proactive methods to maintain and ensure a better quality of service (QoS). In this project, we aimed to try different deep neural network (DNN) methods to predict internet traffic. We take previous utilization of historical network traffic data and other relevant information to predict future network traffic.

To begin with, we researched previous methods for network traffic prediction including traditional statistical models. The first model is the Autoregressive Integrated Moving Average(ARIMA) model. The Autoregressive Integrated Moving Average(ARIMA) model is used to evaluate the impact of large-scale health interventions, especially in the presence of seasonality and autocorrelation. It is a useful tool to evaluate the impact of large-scale interventions when other approaches are not suitable, as it can account for underlying trends, autocorrelation, and seasonality and allows for flexible modeling of different types of impacts. The second model is Exponential Smoothing(ES), a forecasting method for univariate time series data, which produces forecasts that are weighted averages of past observations where the weights of older observations exponentially decrease. Forms of exponential smoothing extend the analysis to model data with trends and seasonal components. By adjusting parameter values, analysts can change how quickly older observations lose their importance in the calculations. Those approaches have low computational costs compared to newer methods using machine learning(ML). However, these methods are struggling to handle data bursts and non-linearity in more complex networks.

Moreover, for more advanced methods like ML, various models have been examined in network traffic prediction. First of all, the Support Vector Regression(SVR) method was devised for single-output regression. Uses a quadratic program (QP) to obtain the predictions of a single output. Support vector regression (SVR) is characterized by the use of kernels, sparse solution, and VC control of the margin and the number of support vectors, and has been proven to be an effective tool in real-value function estimation. The main advantages of SVR are that its computational complexity does not depend on the dimensionality of the input space, and it has excellent generalization capability, with high prediction accuracy. In addition, the Decision Tree (DT) is a tree-like structure composed of internal nodes and leaf nodes. Internal nodes represent decision rules based on features, while leaf nodes represent the output results of classification or numerical values in regression tasks. DTs are constructed through recursive partitioning, with the goal of building a model that performs well on training data and can also make accurate predictions on new, unseen data. Last but not least, genetic programming(GP), is only used to produce automated computer programs by knowing the general concept of the problem and without coding. New generations are delivered by expelling branches from one tree and implanting them in another tree. This basic process assures that the new program is still a tree that is structurally valid. The GP will then perform an evolutionary search operation in a very large space of newly created programs. Those ML methods are superior in solving nonlinear patterns while they still need feature selections and parameter tuning.

In this project, we will mainly focus on using DL algorithms like RNN, LSTM, ConvLSTM, and GNN to better solve the problem of nonlinearity. The knowledge of those DL algorithms will be the foundation of the project.

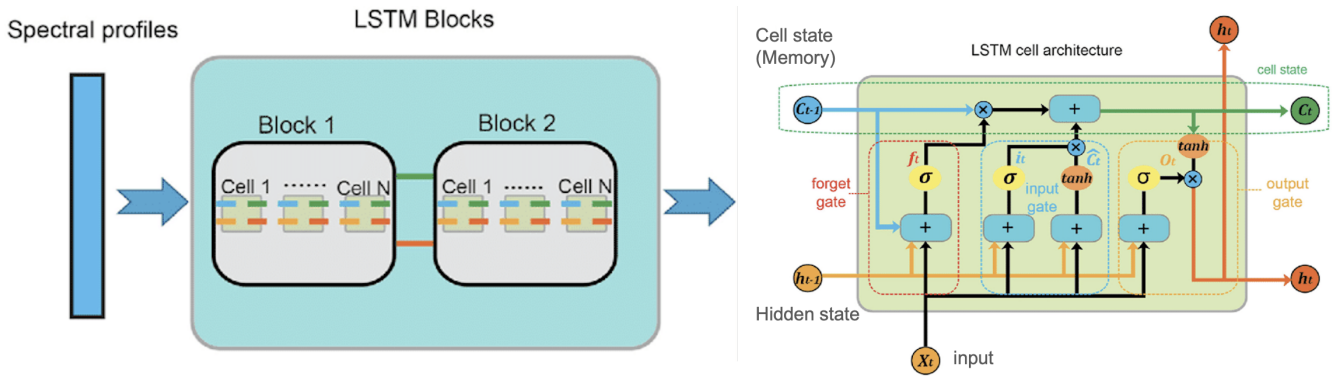


Figure 1: LSTM structure

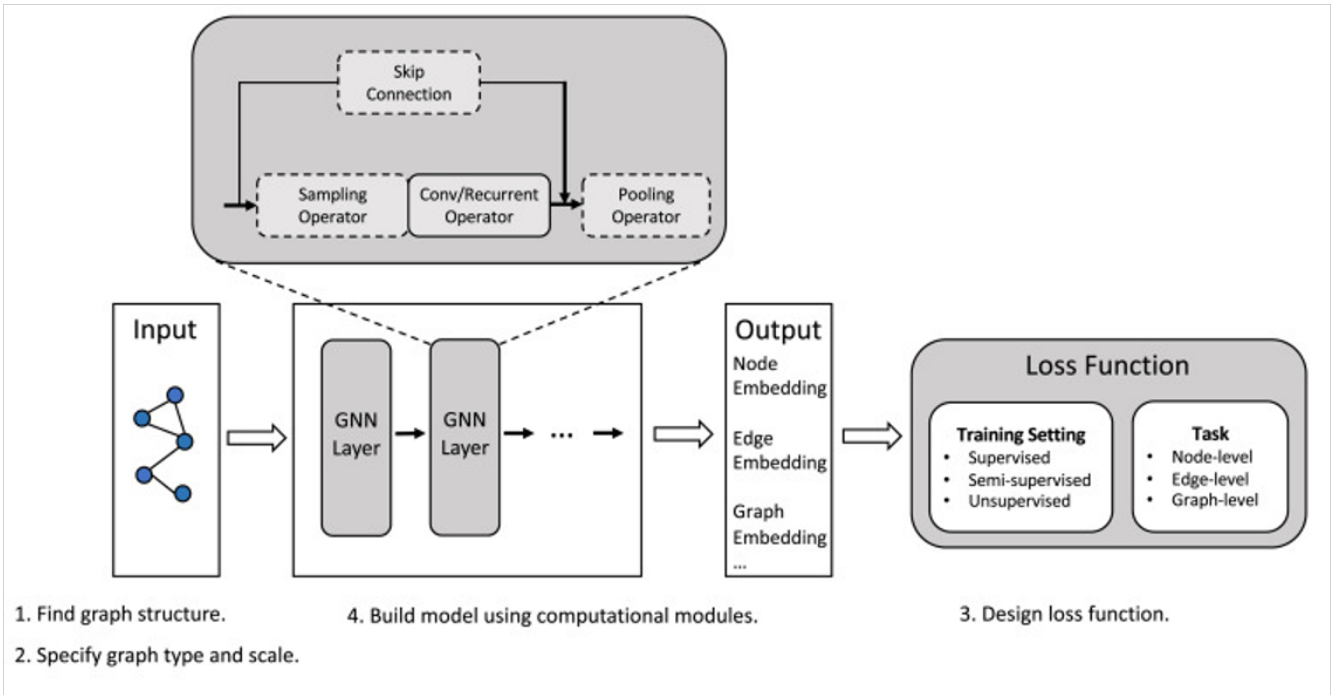


Figure 2: GNN structure

1.1 Traditional deep neural networks

In the traditional deep neural network, LNTP is based on Long-Short Term Memory(LSTM) online traffic prediction model. This model uses wavelet transform to separate data into different frequency components. Then put these components into the LNTP model for training. According to this situation, it can improve the network traffic prediction accuracy including the burst of traffic.

Another DL algorithm is ConvLSTM, which is more good at dealing with spatiotemporal data than LSTM. The paper [12] shows that Spatial-Temporal Cross-domain Neural Network(STCNet) uses Convolutional Long Short-Term Memory (ConvLSTM) components to fetch history traffic data spatiotemporal dependency and continue information. STCNet is a combination of space and temporal features to modeling in order to precisely capture and predict complex nested data patterns.

The other DL-algorithm is the Gated Recurrent Unit (GRU), which is the simple version of the Long Short-Term Memory (LSTM) network. The paper [14] shows that GRU and LSTM are attributed to the extension of Recurrent Neural Networks (RNN). The difference is that GRU combines the forget gate and input gate of LSTM. And GRU doesn't need the separate memory cell state in order to make its structure simpler and its computation efficiency higher. The paper [14] compares the LSTm and GRU in several sequence modeling assignments. They find that GRU performance is better than

LSTM since its simple structure and high performance computation. Moreover, paper [13] verifies the prediction performance by using GRU's Diffusion Convolutional Recurrent Neural Network (DCRNN). DCRNN combines Graph Convolutional Networks (GCN) and GRU in order to deal with spatiotemporal sequence prediction problems in the graph structure data processing. Furthermore, DCRNN's accuracy and recall are better than other models in prediction network congestion. The paper [13] also mentions that DCRNN has the strength of capture network topology structure. Compared with the traditional time sequence prediction model, DCRNN can use graph convolution operations to deal with the complex relationship between network nodes. Then it can provide a more precise prediction result.

1.2 GNN

Graph neural networks (GNN) are neural models that capture the dependence of graphs via message passing between the nodes of graphs. GNNs are deep learning-based methods that operate on the graph domain, which has become a widely applied graph analysis method recently. First of all, Graph neural networks (GNNs) can be used to learn from graph data. GNNs use a message-passing mechanism to aggregate information from neighboring nodes, allowing them to capture the complex relationships in graphs. GNNs are effective for various tasks, including node classification, link prediction, and clustering. In addition, GNNs facilitate the exchange of information between nodes in a graph, enabling them to understand dependencies within the nodes and edges. The general design pipeline of GNNs contains four steps including finding graph structure, specifying graph type and scale, designing loss function, and building a model using computational modules. The array of Python libraries supported GNN models PyTorch Geometric and Deep graph library which can be a foundation for the project programming part. Specific types of GNN include TSGAN, which can integrate spatial similarity information from both time series and network topology in the form of a graph, extracting spatial-temporal dependent features of cellular traffic, and Spatial-temporal Dynamic Graph Network (SDGNet), which combines dynamic graph spectral convolution and diffusion graph convolution to faithfully capture the spatiotemporal correlations of mobile traffic for high- fidelity prediction.

1.3 Transformer

Previous models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. In 2017, Google Brain proposed a simple network architecture, the Transformer, which eschews recurrence and is based solely on attention mechanisms, and also draws global dependencies between input and output[16]. The method has been highly effective in the fields of text processing, natural language processing, and beyond. The core idea is the self-attention mechanism, which allows the model to weigh the importance of different words in the text. Furthermore, the method also adopted parallel running of the self-attention mechanism, contributing to capturing different semantic and syntactic relationships in the data. After the attention mechanism, the data goes through a feed-forward neural network. Since it uses the attention mechanism, the Transformer shifted from sequential data processing to positional encoding. The spatial-temporal downsampling neural network (STD-Net) was introduced to predict the mobile traffic across an entire city [17]. The method splits the city into various parts and focuses on dynamically and simultaneously exploiting the temporal, local, and global spatial dependencies of small parts to reduce the computational complexity. The Spatial-Temporal Transformer (ST-Tran) proposed a method for cellular prediction [18]. ST-Tran has a temporal transformer block to learn the temporal features of every grid in a network by slicing its traffic flows during recent and periodic time intervals. The spatial features are also linked to the information in related grids when generating spatial predictions. Then the two aspects are merged together to form a final prediction.

2 Theoretical analysis

The main part of the project is on model evaluation, where the prediction performance of network traffic focuses on the difference between actual observed values and predicted values. The are four most commonly used metrics and will be utilized in our project, which are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE). MAPE is a percentage, where a smaller value indicates better prediction performance, while the other metrics highlight the quality of prediction performance mainly when comparing multiple models. Assuming yi represents the true value, yˆi represents the predicted value, and N is the total number of observations used for evaluation, the calculation formulas for the four metrics are as follows:

Furthermore, instead of predicting the performance of network traffic, model evaluation also needs to consider the network environment and requirements, balancing time and computational costs, and formulating personalized evaluation criteria. To take into account those factors, network traffic exhibits complex characteristics such as temporal dynamics and bursts. To avoid conceptual drift caused by changes in traffic trends over time, leading to a decrease in predictive accuracy, it is necessary not to use the same prediction model for an extended period. Continuous model updates are required to adapt to new traffic variations. Therefore, the sustainability of model learning should be one of the considerations in

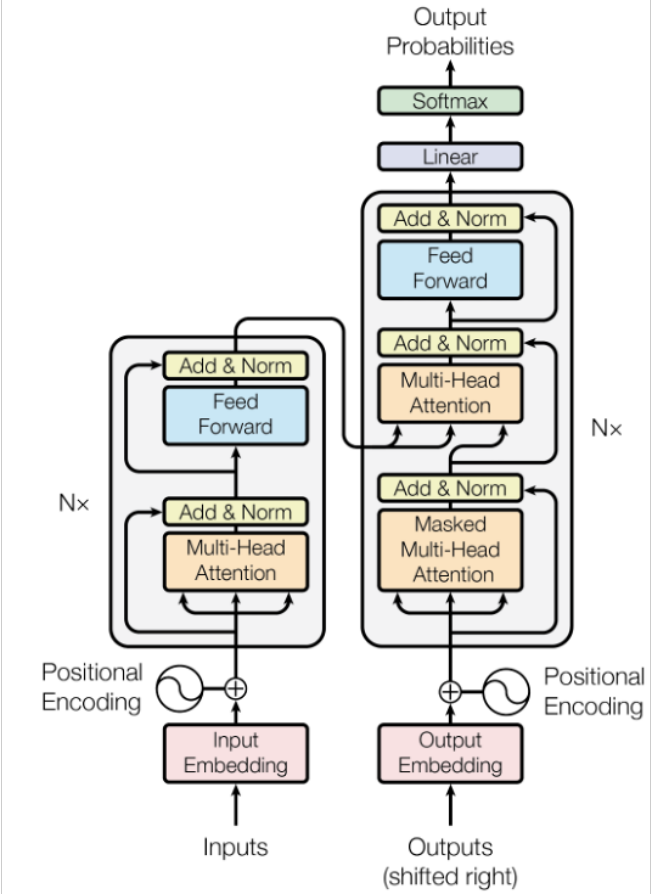


Figure 3: Transformer Structure

model evaluation.

3 Project Goals and Methodology

Nowadays, accurately predicting network traffic in highly interconnected environments is very important. Because it can maintain the network quality and stability. Traditional network traffic prediction methods, such as statistical models, can't deal with burst events and non-linear behaviors even though they can address linear traffic patterns. Our project will propose a new deep-learning architecture to deal with such problems. Since DL algorithms entail a large number of computational parameters, which will cause higher computational costs, our focus will be on how to balance the accuracy and computational cost. If the process takes too long or costs too much, it is not an applicable method to be introduced to the industry.

3.1 Design architecture

We will use deep learning architecture to improve the accuracy of single-step and multi-step network traffic prediction. Deep learning architecture includes the following components:

Dataset: We use online open real-world datasets to implement our project. The datasets we plan to use are Telecom Italia.

Deep Learning Model Selection: We will use LSTM, GNN, and Transformer to analyze the same real-world datasets. We will compare the results and show the superiority or shortcomings of different methods.

Model training and optimization: Utilize end-to-end training strategy to automatically capture important features and optimize modeling parameters by training.

Evaluation: Consider prediction accuracy(eg. MSE, MAPE) and focus on model computing efficiency. Also, conduct an online learning mechanism to update the model immediately.

3.2 Implementation

The dataset had a total of 62 data files, we used the latest one "sms-call-internet-mi-2014-01-01.txt" in our project. The whole dataset had 4030163 entities and after eliminating the null entities, there were 2027304 entities left. We chose the first 10000 data for example. We used the 'pandas' to read the txt files and add the column name including 'gridID', 'timeInterval', 'Internet', etc. Then, We converted the 'timeInterval' column from millisecond to datetime format. Each row of the data is the average internet traffic within ten minutes. We dealt with data within the same 'gridID', which represented they were from a specific area, just like the postcodes in the US. We then normalize the 'Internet' column of data between 0 and 1 using MinMax or calculate it with mean and standard deviation. The data training function defines the ratio of training dataset and testing dataset to be 80 to 20, to ensure a large database and increase model accuracy. We used MSE to calculate the training loss, which determined the fitting rate of the model. Also, we use MSE to calculate the testing accuracy. After training and testing, we plotted the predicted data and original data in the picture. All experiments were run on our personal computers.

3.2.1 LSTM

We used the sliding window technique to create the training dataset and testing dataset that is suitable for the LSTM model. The sliding window size depends on the 'look_back' parameter setting since this parameter defined how many previous data points did they used in each prediction, then we reshaped the dataset into an LSTM model required format such as [sample, time steps, features]. Later, we created a model with

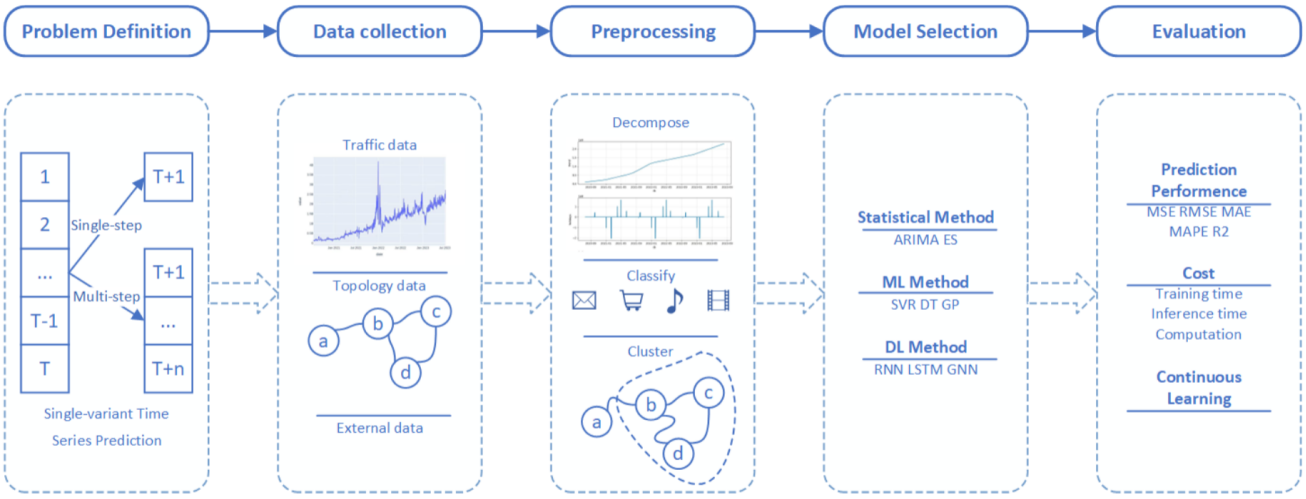


Figure 4: The overall architecture for network traffic prediction

$$MSE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (1)$$

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (2)$$

$$MAE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (3)$$

$$MAPE(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \quad (4)$$

Figure 5: The calculation formulas for the four metrics

two LSTM layers and one dense layer and compiled the model. Then, we used 'adam' optimizer and the mean square error loss function to calculate the loss mean square error(MSE). Moreover, we also trained the LSTM model and set 'verbose=1' to show more detail about the training process. In the end, we used the LSTM model to predict the internet traffic. Then, we converted the prediction result and original data into its original scale by using denormalize. We used the test-predicted data and mean square error function to calculate the test mean square error(Test MSE).

3.2.2 GNN

The GCN function defines the GCN model with two convolutional layers. The graph function constructed the graph data of the dataset. The main function loads data from the CSV file, defines the output, and constructs the graph. The model separately calculated the Loss MSE and test MSE. For Loss MSE, the train loader is created to load the training data in batches of size 32. For test MSE, the test loader is created to load the test data in batches of size 32. The Loss MSE is calculated during each squareID by computing the MSE between the model's predictions and the actual values. This is done iteratively, with the model updating its parameters to minimize the loss. The test MSE is calculated after training by using the model to make predictions on the test data and then computing the MSE between these predictions and the actual values. This indicates how well the model generalizes to new data and the ability to predict. The model plots the true and predicted internet usage data for each Square_ID, which is useful for model selection and result comparison. The code utilized PyTorch and PyTorch Geometric to handle graph-structured data and to do data validation, process, test, and training. Furthermore, the model ensures time interval translates to readable datetime format. The ReLU activation function in the GCN model increases the nonlinear capability of the model. The look_back parameter defines the length of historical data considered by the model at each time step. As shown in the result, the model performs a good prediction as in the Test MSE.

3.2.3 Trandformer

We created the class 'NetworkTrafficDataset' to handle the creation of a dataset suitable for training; the class 'TransformerTimeSeries' to define the transformer model for prediction; and the function 'create_mask' to create masks needed in the transformer. Then we created a training loop and testing loop to train the dataset and make predictions. The 'NetworkTrafficDataset' gener-

ated input (x) and target (y) from the raw data, and made sure that each sequence had the same set length. In 'TransformerTimeSeries', there was an encoder and a decoder. It used linear layers to transform input and output dimensions. The 'create_mask' function created a look-ahead mask for the transformer. The mask hid future position in the sequence when predicting. In the training loop, 'src' and 'tgt' are the source and target sequences. 'tgt_input' and 'tgt_output' were created by shifting the target sequence by one position. Masks were applied. The model was trained by minimizing the loss (MSE) between the predicted and actual data. After that, the prediction loop was predicted on the test data using the trained weights. Masks were also applied. Predictions were transformed to the original scale and figures were generated and stored.

4 Results

We separated the dataset with different grids, which helped to demonstrate the difference between areas in the city. The training figures are all appended in the appendix. The training loss and test loss were calculated by taking an average from all data from multiple grids.

	LSTM	GNN	Tranformer
Train_Loss	0.085	0.053	0.7709
Test_Loss	15.716	0.0527	0.8709

The test MSE from each grid is as follows. The MSE is higher than 1 for some data, indicating an inaccurate prediction.

GridID	LSTM	GNN	Tranformer
1	37.7419	Null	1.3980
10	10.0268	Null	1.1903
1000	16.1202	Null	1.3848
10000	56.5565	Null	0.9526
101	32.9171	Null	1.3519
1001	0.3313	Null	0.1072
1002	4.69	Null	0.9309
1003	0.5651	Null	0.0317
1004	3.3819	Null	0.9368
1005	3.0215	Null	1.0069
1006	3.6190	Null	1.0348
1007	1.6835	Null	0.7016
1008	6.5468	Null	1.4924
1009	2.8683	Null	0.6033
1010	3.0924	Null	0.7617
1011	3.1945	Null	0.6974
1012	3.5816	Null	0.9951
1013	3.6199	Null	0.7479
1014	3.6934	0.027	0.6456
1015	3.5422	0.03	0.6126
1016	3.4835	0.03	0.9068
1017	2.8649	0.03	0.6145

5 Conclusion

Network traffic prediction can help both service providers and customers to use the devices and resources better. In this project, we implemented multiple deep-learning models to predict the network traffic, we compared the prediction accuracy between LSTM, GNN, and Tranformer. For small datasets like the one we used in this project, GNN works best, balancing the computation time and the prediction accuracy. For future discussions, we can train larger datasets on GPU or online servers to better study the performance of the models.

6 References

1. Schaffer, A. L., Dobbins, T. A., & Pearson, S. A. (2021). Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: a guide for evaluating large-scale health interventions. *BMC Medical Research Methodology*, 21(1), 58.

2. Frost, J. (2021, May 18). Exponential smoothing for time series forecasting. *Statistics by Jim.*

3. Tran, N. K., Kühle, L. C., & Klau, G. W. (2024). A critical review of multi-output support vector regression. Pattern Recognition Letters, 178, 69–75.

4. Awad, M., & Khanna, R. (2015). Support vector regression. In Apress eBooks (pp. 67–80).

5. Costa, V. G., & Pedreira, C. E. (2022). Recent advances in decision trees: an updated survey. Artificial Intelligence Review, 56(5), 4765–4800.

6. Oliazadeh, A., Bozorg-Haddad, O., Rahimi, H., Yuan, S., Lu, C., & Ahmad, S. (2022). Genetic Programming (GP): An Introduction and practical application. In Studies in computational intelligence (pp. 251–271).

7. Z. W. Mengyi Fu, Pan Wang and Z. Li, "Deep learning for network traffic prediction: An overview," IEEE, 2023.

8. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. AI Open, 1, 57–81.

9. Y. Fang, S. Ergut, and P. Patras, "Sdgnet: A handover-aware spa- tiotemporal graph neural network for mobile traffic forecasting," IEEE Communications Letters, vol. 26, no. 3, pp. 582–586, 2022.

10. Z. Wang, J. Hu, G. Min, Z. Zhao, Z. Chang, and Z. Wang, "Spatial- temporal cellular traffic prediction for 5g and beyond: A graph neural networks-based

approach," IEEE Transactions on Industrial Informatics, vol. 19, no. 4, pp. 5722–5731, 2022.

11. L. Zhang, H. Zhang, Q. Tang, P. Dong, Z. Zhao, Y. Wei, J. Mei, and K. Xue, "Lntp: An end-to-end online prediction model for network traffic," IEEE Network, vol. 35, no. 1, pp. 226–233, 2020.

12. C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," IEEE Journal on Se

13. D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WORKSHOPS). IEEE, 2019, pp. 246–251

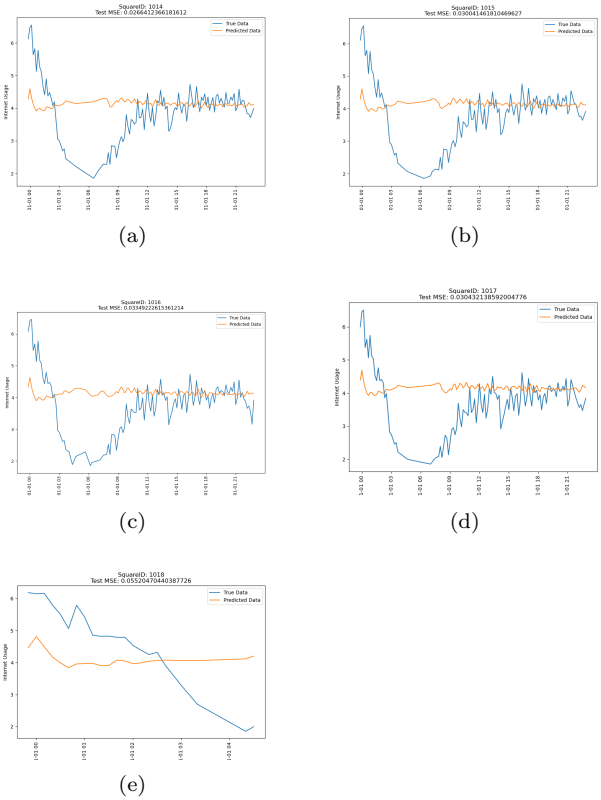
14. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.

15. Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional re- current neural network: Data-driven traffic forecasting," arXiv preprint arXiv:1707.01926, 2017.

16. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

17. Y. Hu, Y. Zhou, J. Song, L. Xu, and X. Zhou, "Citywide mobile traffic forecasting using spatial-temporal downsampling transformer neural networks," IEEE Transactions on Network and Service Management, vol. 20, no. 1, pp. 152–165, 2022.

18. Q. Liu, J. Li, and Z. Lu, "St-tran: Spatial-temporal transformer for cellular traffic prediction," IEEE Communications Letters, vol. 25, no. 10, pp. 3325–3329, 2021.

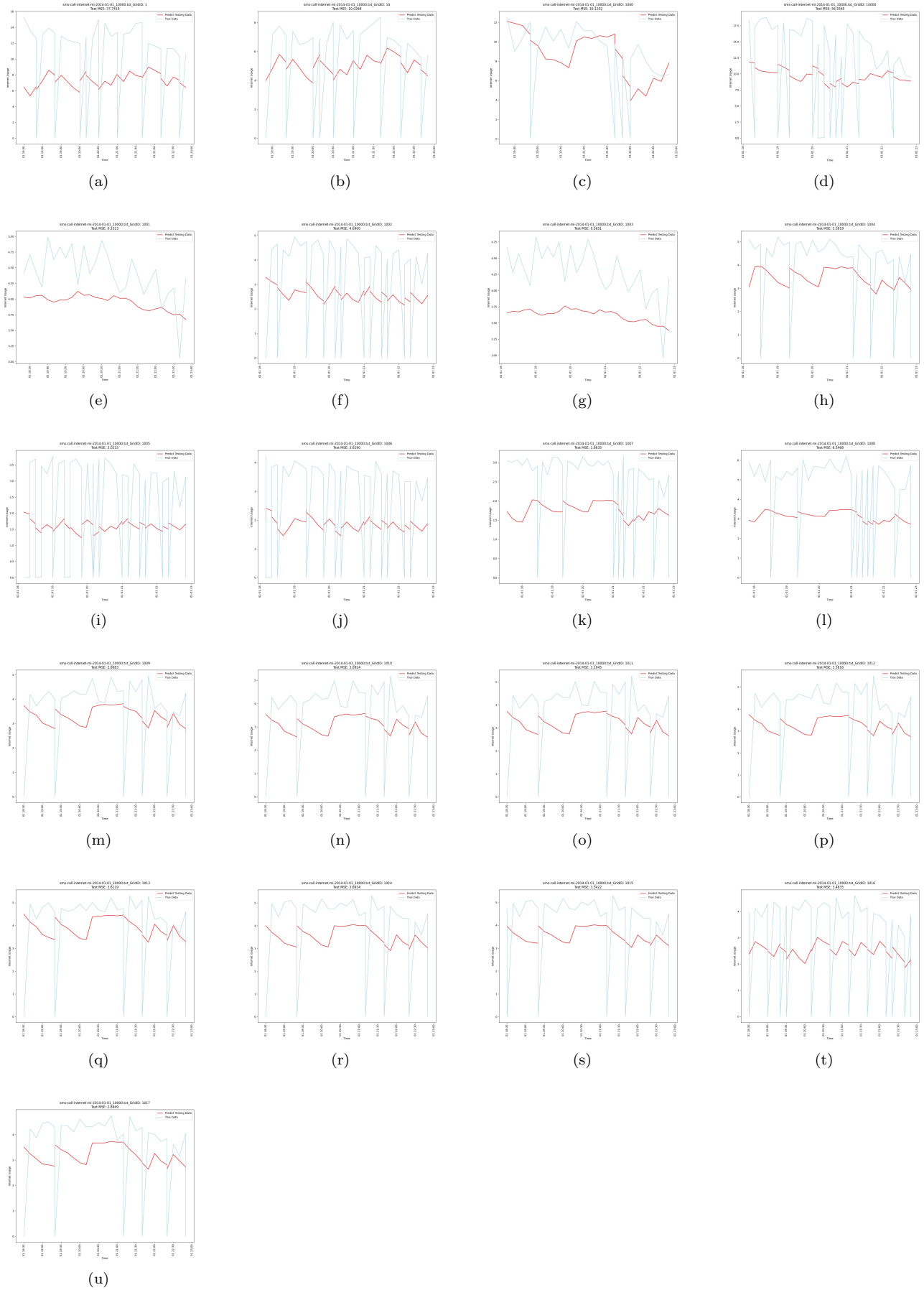


(a) (b) (c) (d) (e)

Figure 6: GNN prediction results

7 Appendix

Github repository://github.com/HowardTsai6/EEC273_final-project



(a) (b) (c) (d)

(e) (f) (g) (h)

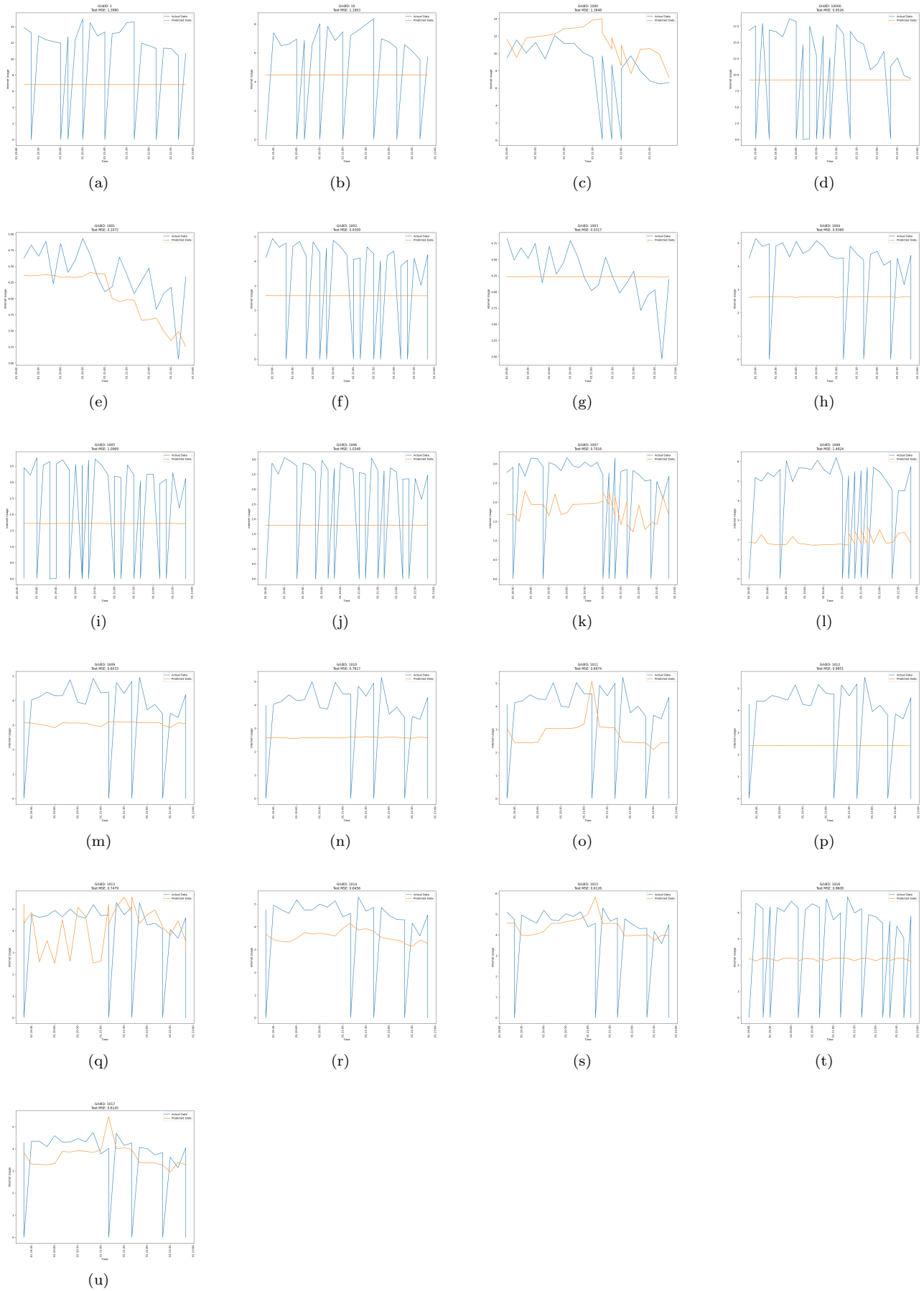
(i) (j) (k) (l)

(m) (n) (o) (p)

(q) (r) (s) (t)

(u)

Figure 7: LSTM prediction results



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

(q) (r) (s) (t)

(u)

Figure 8: Transformer prediction results