School of Computer Science
The University of Auckland
New Zealand

Weather-Informed Traffic Flow Forecasting: Integrating Meteorological Data into Local Spacetime Neural Networks

Yatai Tian

June 2023

Supervisor: Dr Kaiqi Zhao

A dissertation submitted in partial fulfillment of the requirements of Bachelor of Science (Computer Science Honours)

Acknowledgements

I want to express my gratitude to my supervisor, Dr Kaiqi Zhao, for their support and guidance throughout the journey of completing this dissertation. His expertise in the field of graph neural networks has truly been invaluable, and has contributed significantly to this research paper. I am grateful for their countless hours providing constructive feedback and addressing my queries throughout every meeting.

Furthermore, I extend my gratitude to the entire Department of Computer Science for providing an engaging and stimulating academic environment. I have broadened my knowledge immensely and have always been inspired to challenge myself and think critically. Thank you for the numerous seminars and guest lectures organized by the department, which have kept me engaged in the advancements in the field of Computer Science.

Lastly, I am thankful for Yang Song, who provided me with the intricacies of the computational aspect of this research. Their generosity in providing a quick-start guide to the model I have built upon has been instrumental in the success of this dissertation.

Abstract

Traffic prediction is a critical area in forecasting for efficient traffic management and planning. Recent advancements in traffic prediction have yielded promising results, with models getting much more sophisticated every year. However, unlike state-of-the-art models, which typically only include spatial and temporal dimensions, we propose a novel approach incorporating weather patterns as part of traffic prediction. We argue that weather data plays a pivotal role in traffic flow and use data such as rain and the condition of the area as additional variables to predict traffic flow better. We evaluate our approach on a real-world public dataset and show that our experiment leads to an improvement in the overall short-term accuracy of traffic flow prediction. Specifically, our approach leads to improved forecasting if the model were only to take in spatial and temporal dimensions. The improved model has practical implications, as it can assist transportation algorithms in making better-informed decisions to increase traffic flow and reduce congestion in cities.

Contents

1 Introduction **1**

2 Literature Review **5**

3 Methodology **9**

4 Experiments **17**

5 Conclusion **27**

List of Figures

1.1 Traffic Network into Nodes 3
1.2 Spatial Temporal Dimension 3

3.1 PeMS Traffic (Left) and Airport Weather Data (Right) 12
3.2 Reverse Engineering Input Data 13
3.3 Smoothed Reverse Engineering Input Data 14
3.4 Frequency of Group Lengths of 1 15

4.1 Temperature and Dew Point column 19
4.2 RMSE for 15 min Forecast 22
4.3 RMSE for 60 min Forecast 22

List of Tables

2.1 Model accuracy with PeMS dataset 7

4.1 Baseline Results for STNN shortened data 18
4.2 STNN with Temperature 18
4.3 STNN with Dew Point 19
4.4 STNN with Dew Point - Closest Sensor 19
4.5 STNN with Wind Speed 20
4.6 STNN with Light Rain 20
4.7 STNN with Poor Conditions 21
4.8 Original STNN 21
4.9 STNN 15% 21
4.10 GMAN and Modified Inputs (RMSE) 23
4.11 1% increase, 2% decrease 23
4.12 2% increase, 2% decrease 23
4.13 Original STNN - Full Dataset 23
4.14 5% increase, 5% decrease 24
4.15 10% increase, 10% decrease 24
4.16 Exponential increase, Exponential decrease 24
4.17 Overall performance of short-term (15 mins), mid-term (30 mins) and long-term (60 mins) traffic forecasting 25

1

Introduction

Traffic prediction has been a key challenge as part of the Intelligent Transportation Systems (ITS), and a vast number of researchers and engineers have been committed to improving the accuracy of traffic prediction models [16]. There have been significant changes to traffic prediction models, specifically data-driven approaches towards traffic prediction. With methods starting from simple time series to present day machine learning and artificial intelligence, traffic prediction models have evolved tremendously and there are now many different models available.

Despite the progress made in previous years, the complexity and dynamic nature of traffic networks combined with other factors, including weather conditions, unpredictable vehicle accidents and road construction detours, make achieving high accuracy levels challenging. While current-day models have shown encouraging results, it is clear that there can still be a long way to go in future advancements of traffic prediction models.

The average United States driver spends roughly 22,000 kilometres on the road each year [5]. Given this sheer volume of travel, it is paramount to refine accurate traffic prediction. There are many potential benefits of having better traffic predictions, notably reduced travel time, improved safety, and better traffic flow. Predicting the flow of traffic allows systems to guide drivers along less congested routes, therefore saving time and improving the efficiency of travel along the roads. Fewer cars on a stretch could also lead to improved safety as the risk of crashing into another decrease. This makes it an important area of research and development for the future of ITS.

A traffic network can effectively be modelled as a graph, where the nodes in the graph represent the many traffic sensors, and the edges represent the road network between them. Take the representative example below in figure 1.1, where many traffic sensors are scattered across Victoria Park. Each sensor here collects information needed for analysis, such as the flow rate of traffic. The information these traffic sensors hold can be stored and represented under the corresponding nodes in the graph model. Edges can be constructed by looking at the road network, and information such as the distance between edges can also be stored. The spatial dimension is not the only information recorded, as we also incorporate the temporal dimension. This is because we use historical speed data to predict the speed at a future time step. This temporal dimension can be shown in figure 1.2, where we also capture time data. We are using the traffic flow rate in T_n and all preceding timestamps up to T_{n-k} and using these data points to predict the traffic flow rate for future timestamps T_{n+1}. Existing machine learning models, specifically graph neural networks (GNNs), have performed well in modelling traffic rate and has been a significant step in accurately representing non-linear traffic features. GNNs capture traffic data's

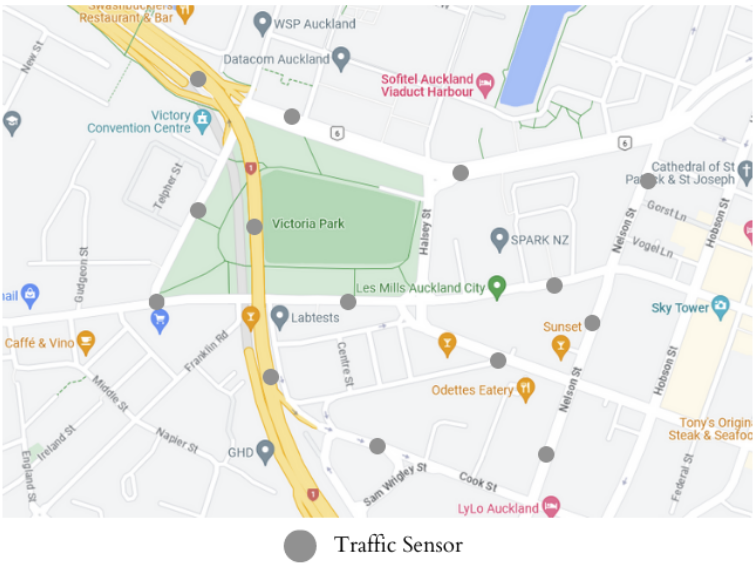

Figure 1.1: Traffic Network into Nodes

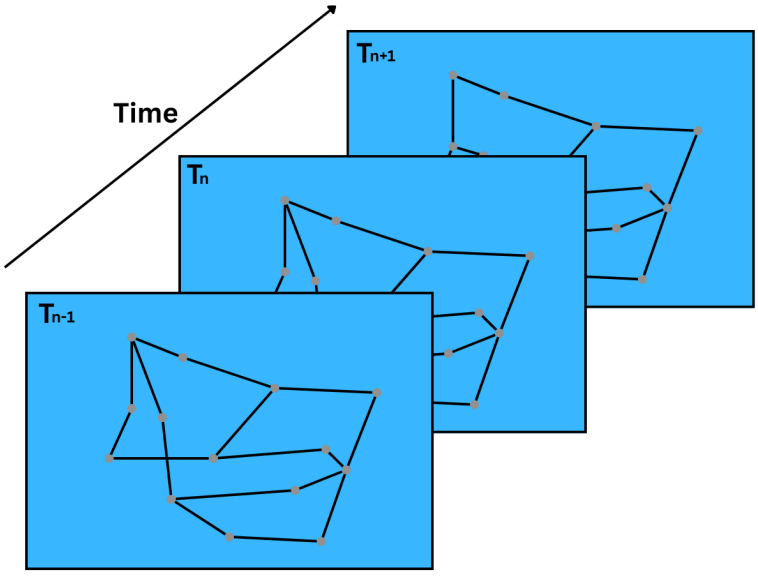

Figure 1.2: Spatial Temporal Dimension

spatial dimension and perform much better than classical statistical methods. However, GNNs depend on the graph's structure, which means moving to a different road network would result in a retrain of the model. GNNs are also computationally expensive, where they demand feature aggregation operations on the entire network and scale quadratically in relation to the number of nodes. To combat these limitations, we will use a Space-Time Neural Network (STNN), which combines attention blocks and space-time convolutional blocks that can handle dynamic networks and has a complexity independent of the network size.

The current state-of-the-art STNN model has great accuracy but misses features such

as meteorological data, which we argue is pivotal in forecasting the rate of traffic. Weather conditions heavily influence driving behaviours which will have an effect on forecasting. Driving in rainy weather means less visibility on the road ahead due to all the rain hitting the windshield and the foggy conditions associated with rain. The tyres also have less grip as the rain could cause slippery surfaces and hydroplaning. These factors would tend to result in drivers reducing their speed and slowing down.

Not only does weather impact visibility and road conditions, severe thunderstorms and flooding can see road closures or diversions, which plays into how complex modelling traffic data is. Drivers will most likely leave earlier than usual, seeing poor weather, which also alters expected traffic patterns. Depending on the number of drivers sharing the road, this can easily slow down traffic flow and introduce phantom traffic jams. Phantom traffic jams occur when drivers suddenly break, which causes a ripple effect backwards through traffic. This effect can be seen for a considerable amount of time even though the driver that caused this is no longer to be seen.

Existing weather approaches focus on models that have limitations in capturing all the complexities of real-world traffic flows. The models used include ANOVA, conditional regression methods, and CNN-LSTM models. The models also use data that is not highly used in competitive machine learning models and compare those with basic statistical methods.

Meteorological data is not just an additional variable in traffic prediction, but it is clear that it is a pivotal aspect in the rate of flow traffic prediction. We incorporate such data into the state-of-the-art STNN model and show it enhances its predictive accuracy and thereby providing a more comprehensive overview of traffic patterns.

The structure of the remaining paper is outlined as follows; related research and methodology are introduced in Sections 2 and 3, respectively. The findings are presented in Section 4 and a discussion of the results is provided too. Finally, the paper concludes with a summary of the key findings in Section 5.

2

Literature Review

In this section, we discuss the most relevant prior work on the problem of traffic forecasting. We discuss early models from the late 1900s to current state-of-the-art methods. These can be nicely summarised into four sections of work.

Statistical. The early days of traffic prediction were narrowed down to variations of a time series model. Techniques used included historical averages (HA) and autoregressive integrated moving average models (ARIMA). Williams *et al.* proposed that seasonal ARIMA models (SARIMA) outperformed exponential smoothing models [14]. Zivot showed vector autoregression (VAR) was proven helpful in describing the dynamic behaviour for time series and forecasting [23]. Okutani *et al.* proposed two models for predicting short-term traffic movement based on the Kalman filtering theory [11]. Kalman filtering was used to estimate real-time traffic conditions based on current traffic measurements from various sensors. This was an improvement from prediction algorithms that used historical data as a reference. However, these statistical models rely on the assumption that traffic data is weakly stationary, which is not the case for complex networks.

Machine Learning. Machine learning algorithms started to outperform statistical methods in the early 2010s. Lippi *et al.* showed that several machine learning algorithms, such as support vector machines (SVM) and decision trees, outperformed traditional statistical methods in terms of prediction accuracy and robustness [9]. Chen *et al.* proposed a forecasting model using seasonal support vector regression (SVR) [2]. They showed that their model outperformed several traditional time series predicting models, including seasonal ARIMA and exponential smoothing models. Combined with the SVR was an adaptive genetic algorithm that optimised model parameters and improved the model's generalisation. These methods improved on statistical methods but failed to capture complex nonlinear relationships of traffic data.

Deep Learning. Deep learning methods such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) started to gain traction and outperform traditional machine learning methods from about 2015 onwards [7]. These models allowed for capturing more complex traffic data and finding intricate patterns in the data. Zhang *et al.* proposed a deep learning model based on long short-term memory (LSTM) networks for predicting traffic flow. LSTM networks are able to learn complex temporal patterns in the data, and Zhang showed that this outperformed traditional machine learning and statistical methods. Zhao proposed a gated recurrent unit (GRU) method, a variant of the RNN structure [21]. Zhao utilises data fusion and enhances the input data's quality by combining different data sources, which include historical travel time data, real-time travel time data, and weather data. The findings show that the GRU-based model with data fusion performs better than older techniques of statistical and machine-learning methods in terms of prediction accuracy. More specifically, for the well-known dataset of California's Performance Measurement System (PeMS), the GRU-based model achieves

a mean absolute percentage error (MAPE) of only 8.87% and a root mean square error (RMSE) of 1.37 minutes. Comparing this to the older methods of ARIMA, SVR and LSTM can be summarised in the following table 2.1. These methods do not capture any

Model	MAPE (%)	RMSE (mins)
GRU	8.87	1.37
ARIMA	24.26	2.66
SVR	18.21	2.32
LSTM	10.11	1.49

Table 2.1: Model accuracy with PeMS dataset

spatial component which is pivotal to any traffic forecasting.

Graph Neural Networks. Graph neural network methods built onto what deep learning has to offer and started to outperform them in 2017. While neural networks were great for time series, the spatial component is just as crucial in making predictions. GNNs can handle spatial relationships between road networks and therefore provide a better understanding of traffic flow dynamics. We include both spatial and temporal information to predict the rate of traffic better. Li *et al.* first introduced this in data-driven traffic forecasting, where they use a diffusion convolution layer to capture spatial dependencies with random walks and a sequence to sequence architecture [12] that captures temporal data [8]. Yu *et al.* proposes a graph convolutional network (GCN) to model relationships between road structures and a temporal convolutional layer plus gated recurrent unit to capture long-term temporal dependencies in the traffic data [20].

2.0.1 Incorporating Weather

Nigam *et al.* introduce a hybrid RNN-LSTM model incorporating weather impact for prediction [10]. The paper acknowledges that weather conditions such as fog, snow and rainfall all significantly impact driver speeds due to the reduced visibility of drivers. They also note that heavy rain can cause substantial water-logging in developing countries due to their weak drainage systems, which results in a prolonged impact on traffic conditions. Similarly, heavy snow can create thick layers on the road affecting the rate of traffic flow. They tested this on a San Diego dataset. Specifically for the I-5 road, the traffic flow prediction error of the LSTM model was 2.3% with the rain data and 6.54% for the model without rain data for a 15-minute prediction horizon.

Hou *et al.* aims to tackle short-term traffic flow with respect to the influence of weather conditions [6]. Hou uses a combined model of a stacked autoencoder (SAE) and

radial basis function (RBF) neural network to predict the rate of traffic. The process uses much data preprocessing work with generating HA models, one-hot encoding, embedding components, and PCC and PCA being applied to feature-level data fusion for weather parameters. Overall, they showed that their model performed better at all time points before 30 minutes, comparing this to Artificial Neural Networks (ANN) [1], Gated Recurrent Unit (GRU) [3], Long Short-Term Memory (LSTM) [13] and many others.

Dune *et al.* introduces a stationary wavelet transform (SWT) which holds information on rainfall data into the traffic prediction model [4]. They argue that traffic prediction algorithms should take rainfall into account. They find that rainfall and traffic volume do not have any immediate correlation, but it is visible at specific frequency levels. They use rainfall as an exogenous variable and compare RMSE accuracy of 125.42 to an ANN model with a RMSE of 165.89. These papers show the necessity of weather-adaptive traffic prediction algorithms.

2.0.2 STNN Model

Yang *et al.* shows that we need to understand both the extrinsic and intrinsic factors that influence traffic patterns to accurately predict the flow of traffic [19]. They propose an innovative learning paradigm called Spacetime Interval Learning, which fuses spatial and temporal dimensions. The model combines attention blocks, which help prioritise relevant features, and convolutional blocks, which help capture local data patterns. One of the distinguishing aspects of the Spacetime Interval Learning model is that it does not rely on a specific graph structure, making it universally adaptable to a range of other graph contexts. The STNN model significantly outperformed pre-existing methodologies in the short-term forecast and outperformed some methods in the mid-term (30-minute) prediction. However, the model misses the necessity of incorporating weather patterns as a feature, as it plays a crucial role in influencing traffic conditions. This paper aims to delve deeper into capturing the weather patterns and integrate them as a significant feature in the traffic prediction model. The weather data should enhance the model's predictive accuracy and provide a better understanding of how weather influences traffic.

3
Methodology

This research project will focus on experimenting with a pre-existing local space-time model designed by Yang *et al.* The main steps include gathering and formatting weather data to work with the model and playing around with the different methods of incorporating the weather data. These methods include weighted averages with weather sensors, using the closest sensors and using softmax distances. The weather data was also added as an extra variable, and also as a change to input data.

One real-world dataset was used for evaluation, the PeMS-Bay dataset. This is because extracting weather information from the other common dataset used in traffic prediction, METR-LA, showed zero or extremely low rain history in the region. This meant no new information was included in the model and therefore was scraped as it was not helpful towards our research. PeMS-Bay is extensively used for traffic prediction models and is also used here to make a fair comparison with the model's performance towards existing approaches. PeMS-Bay contains data from 325 sensors in the Bay Area and spans a period of six months in early 2017. The dataset includes speed readings which are recorded and aggregated to five-minute windows resulting in 288 data points per day. Missing values from these readings are filled using linear interpolation.

Weather data corresponding to the timings for PeMS-Bay was scraped from Wunderground with the help of Selenium and the Chrome browser. A few other websites were looked into, but were either paid or had less data to work with. The Wunderground weather data can be found under the history tab with the table name 'Daily Observations'. The table includes ten features, nine of them relating to meteorological data and one of them being time. The scraping process took roughly four hours, with pauses between each day, to confirm the table loads before collecting the required information. In the end, there was only one missing observation seen here for weather station KPAO on June 2nd 2017. This has been replaced with the data from the previous day. Both linear interpolation and extrapolation were used to aggregate meteorological data to five-minute intervals that fit the interval for traffic data, allowing an easier merge process.

The choice of weather variables was investigated, and it was decided whether to be used in the model or left out.

Time. This must be used to merge the two datasets together, but using this as a feature does not make any sense. Time is mostly noted in hourly intervals, with the odd 30-minute and random interval recordings.

Temperature. Variations in temperature may influence driver behaviour, such as extreme high/low temperatures deterring drivers from travelling. However, this is deemed a low-impact variable and would not be used first in the model. Temperature measured

in degrees Fahrenheit.

Dew Point. High dew points may correspond to foggy conditions. This reduces visibility, and hence we would assume slow drivers down. This would be useful in our model and is measured in degrees Fahrenheit.

Humidity. Similar to the dew point, where high humidity may cause foggy conditions. This feature will be considered in our model and is recorded in %.

Wind. The direction of the wind is less useful but may impact high-profile vehicles. This is deemed not useful and not considered for the model.

Wind Speed. + Wind Gust. Wind speed and wind gust record pace and abrupt variations in wind velocity, respectively. Higher wind speeds may make drivers more cautious in driving. This would be useful in our model but not a high priority. Both measured in mph.

Pressure. Pressure may influence other variables to change weather conditions. However, the low change in pressure over time and direct impact on traffic is deemed insignificant compared to other variables and, therefore, not considered for this model.

Precip. The higher the precipitation would mean more rain or snow falling. Higher rainfall would lead to worse conditions on the road, which likely slows down drivers and therefore impacts the rate of flow measurements. This feature would be useful in our model and is measured in inches.

Condition. Great at telling the nearby conditions from the sensor at each time point. Conditions like 'Cloudy' and 'Fair' were deemed minimal as the rate of traffic would not be as heavily impacted as if the condition was 'Thunder' or 'Heavy Rain'. About 20 different categorical conditions were recorded, providing a great perspective on weather patterns that could influence traffic flow. Since this is categorical, we would need to convert this into numerical values for our model.

Multiple experiments were run with a mix of different features for the STNN model. First, we added only one extra feature, such as only temperature, only dew point, and only using humidity. A mix of these three features was also used to see any improvement in the model. For PeMS-Bay data, there are 325 sensors scattered across the Bay Area shown on the left of figure 3.1. From Wunderground, only airport sensors included free historical temperatures and therefore, four of the closest airports were extracted, shown in the right of figure 3.1. The goal here was to find a way to incorporate data into each sensor, and various methods were experimented with. Grabbing the inverse distance was one experimented method, $\frac{1}{\text{distance}}$, and was transformed to using exponentials, so further distances were weighed exponentially less than closer distances. These were normalised, and the weighted value was passed as a new column. For example, let d represent the distances between the desired sensor and the four air-

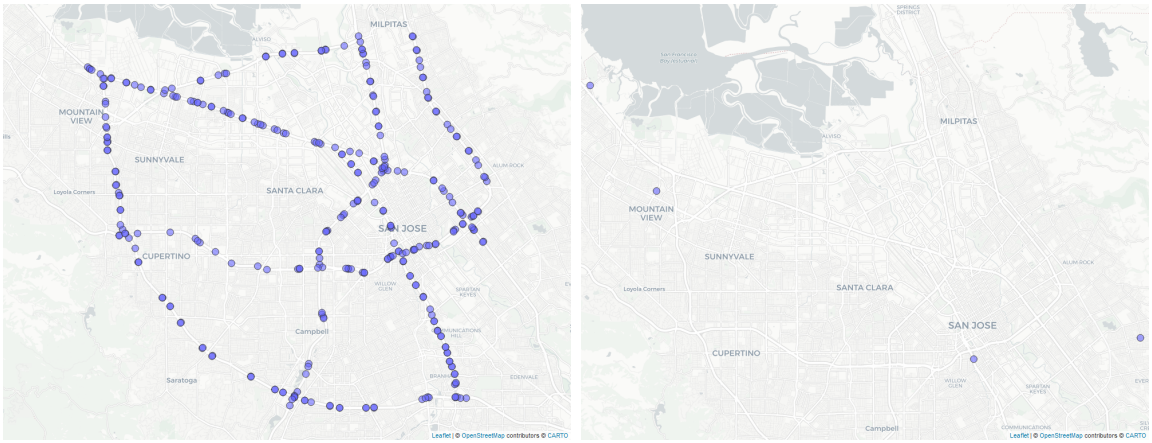

Figure 3.1: PeMS Traffic (Left) and Airport Weather Data (Right)

ports, $d = [1, 2, 3, 4]$. Let the four airports have a certain temperature value t where $t = [100, 70, 80, 90]$. Then, we apply $\frac{1}{\text{distance}^2}$ to get $d' = [1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}]$. Now normalised into $d'' = [0.70, 0.18, 0.078, 0.042]$. We then calculate the weighted temperature and place it in a new column, $100 * 0.70 + 70 * 0.18 + 80 * 0.078 + 90 * 0.042 = 92.62$. Another method experimented with was purely taking the closest airport for 100% of the data. Using the previous example, instead of taking the weighted temperature, we would take the first value of 100 as a distance of 1 would be the closest.

Experiments were also run with the categorical column condition. The condition column included a word to describe the current condition of the area. These words could be but not limited to 'Fair', 'Rain', 'Cloudy' etc. There were around 30 different words, so if we ended up using one-hot encoding, this would stretch to 30 columns which would slow down training tremendously. To combat this, we picked out specific columns that would make the flow of traffic slower so we did not have to use all the columns. These included any condition that included 'Rain', 'Thunder', 'Storm', etc.

One thing noticed was that categories such as 'Storm' only appeared a few times out of all the timestamps, for example, three times out of 52216 timestamps. To fix this, we merged all the columns into one column, 0 if none of the columns have a 1 and 1 if any of the columns have a 1. This made it so we only have one extra feature to use with prediction.

The STNN model is designed to be used for other traffic datasets, ones that it has not seen before. Because of this, instead of adding an extra column, we decided to manipulate the input data and 'reverse engineer' weather patterns. This can be seen in figure 3.2.

This meant increasing the value for the flow of traffic by a certain percentage if there was a rain reading. The figure shows we start with 0's where we have the original data.

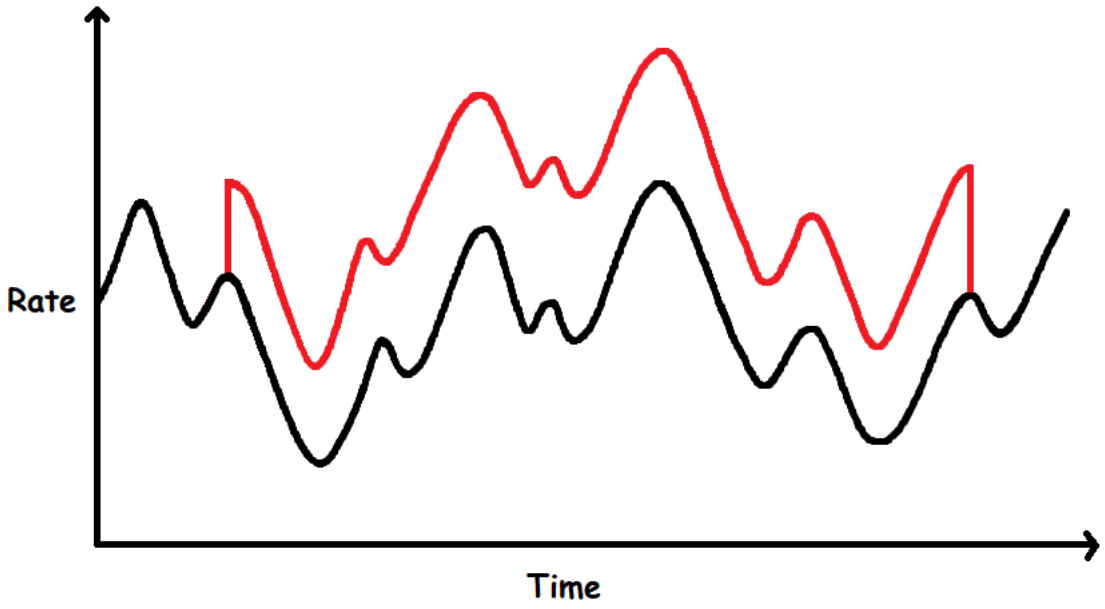

Figure 3.2: Reverse Engineering Input Data

Following this, we have lots of 1's showing poor weather, which increases the input data to the red line. After reading back 0's, we return to the original data. The percentage was played around with and used in experiments. Not only was a fixed percentage used, but a variable percentage increase was used. This is because stray '1' values indicating poor weather may impact the training of the model. Say we had a constant flow of traffic [10, 10, 10, 10, 10], but the binary weather data had a stray '1' [0, 0, 0, 1, 0]. With a 20% fixed increase, we would get our new flow of [10, 10, 10, 12, 10]. The variable percentage was designed to start at 1% and increase by 1% for each consecutive 1. This means an original [10, 10, 10, 10, 10] with a weather array of [0, 1, 1, 1, 1] would produce a new flow of [10, 10.1, 10.2, 10.3, 10.4]. If a 0 followed in the weather array, we would decrease the variable percentage by 2%, up to a minimum of 1%. The maximum percentage was also experimented with, as 100 1's in a row would lead to a 100% increase, something unlikely to happen in real traffic flow. This variable increase can be seen in figure 3.3 with smoothed-out beginnings and ends.

Not only did we work with the STNN model, we manipulated the data the same way with a graph multi-attention network (GMAN). The model was run with the original dataset and then fixed percentage increments to see if the model performed any better.

Furthermore, the sudden percentage increase in traffic may impact the model significantly, so smoothing techniques were applied to reduce jumps in input data. The number of 1's that occurred together can be seen in figure 3.4. From this, we can see that sec-

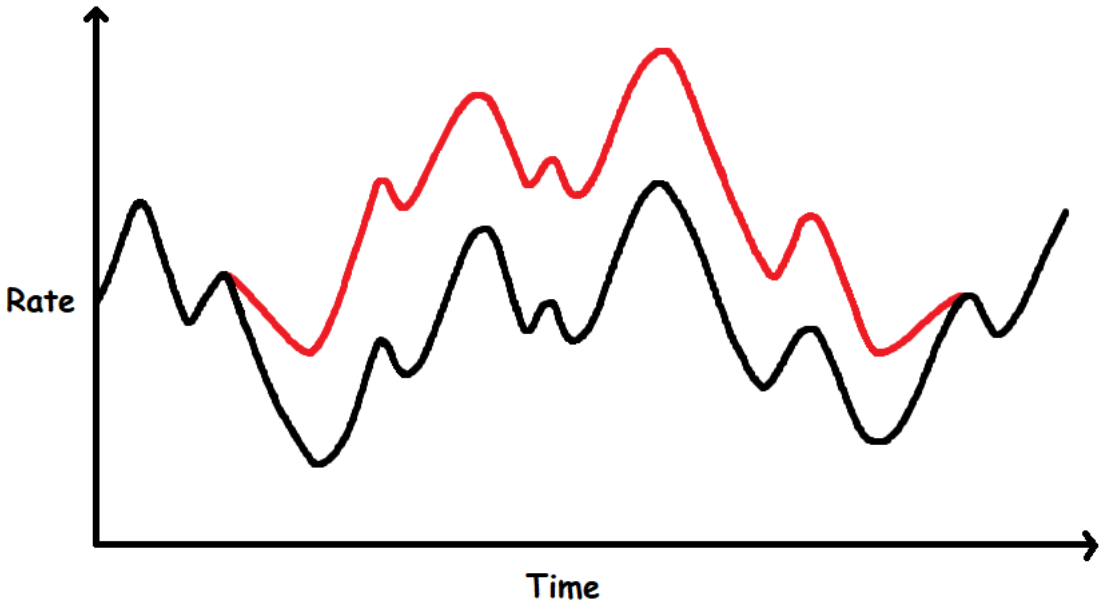

Figure 3.3: Smoothed Reverse Engineering Input Data

tions with many consecutive 1's represent poor weather for multiple days. The graph is right-skewed, meaning shorter group lengths of 1 are more common than longer group lengths of 1. To model this into our input data, we can slowly increase the rate of traffic based on the number of consecutive 1s. This included increasing the rate by 1% for every consecutive 1 occurring and having a max increase cap. Both the increase rate and the max increase cap was tested to find the best result. Linear and exponential functions were also tested with the input data. For example, an increase of 1, 2, 4, and 8% maxing out at 16% for consecutive 1's. If a 0 occurred after a 1, showing that the weather turns better for drivers, the rate decreased by a certain percentage. This percentage was also added to the input data and also tested without adding this percentage.

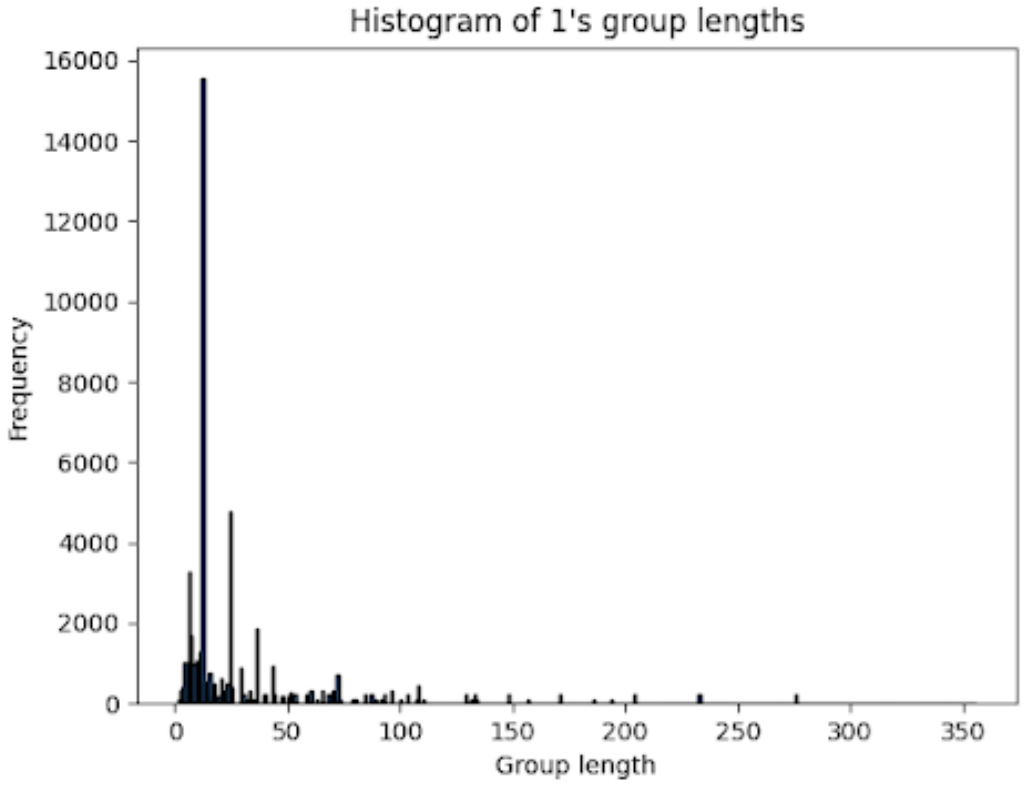

Figure 3.4: Frequency of Group Lengths of 1

4
Experiments

The original STNN model took about 5 hours to train on the METR-LA data with specifications of; CPU - Intel(R) Xeon(R) Gold 6128 CPU @ 3.40GHz, GPU - NVIDIA Quadro RTX 8000. The PeMS-Bay area data includes a longer set of dates, and I did not have access to an RTX 8000. This resulted in much work done by reducing the dataset to $\frac{1}{10}$th of its original size and ran locally before moving onto an RTX 3090 workstation provided by the university. Local models ran overnight on the full dataset only reached step $\frac{7}{50}$, which meant reducing was compulsory for experiments to take place. After shrinking the dataset, the model took around 3 hours to train on an i7-12700KF.

The shortened data consisted of 1212 timestamps for 325 sensors. It was done by computing the mean of equally sized segments in the second dimension. A baseline model was run to achieve the following results in 4.1. Each prediction step is five minutes, so a prediction step 3 would mean predicting 15 minutes later.

Predict Steps	**MAE**	**RMSE**	**MAPE**
3	2.778	5.288	5.808%
6	2.878	5.383	5.956%
9	3.087	5.817	6.534%
12	3.250	6.083	6.908%

Table 4.1: Baseline Results for STNN shortened data

The next step was adding the temperature to the third dimension. The third dimension stored variables time and rate of traffic, so the temperature was added as the third variable. A sigmoid function calculated the temperature value by a weighted average of the four weather stations nearby. The results are shown in 4.2.

Predict Steps	**MAE**	**RMSE**	**MAPE**
3	3.176	5.598	6.605%
6	3.224	5.604	6.624%
9	3.414	6.042	7.167%
12	3.546	6.253	7.448%

Table 4.2: STNN with Temperature

Adding temperature made the model worse by a significant margin. This may be because the temperature has little to no impact on driver speed, and adding unnecessary information caused the model to perform worse. Testing the model with dew point as an additional variable instead of temperature was considered more likely for the model to perform better. This is due to water being a main factor in slowing down drivers. The results are shown in 4.3.

Unfortunately, dew point is seen to perform worse than the temperature. An explanation is that dew point follows a similar trend to temperature but has much less of a

Predict Steps	MAE	RMSE	MAPE
3	3.241	5.530	6.586%
6	3.238	5.577	6.629%
9	3.519	6.188	7.376%
12	3.665	6.476	7.739%

Table 4.3: STNN with Dew Point

standard deviation, as seen in figure 4.1. The values do not change much and therefore are worse for the model. Another experiment was run but using the closest weather station value rather than a weighted average in table 4.4.

Time	Temperature	Dew Point
12:53 AM	61 °F	52 °F
1:53 AM	60 °F	52 °F
2:53 AM	60 °F	52 °F
3:53 AM	59 °F	51 °F
4:53 AM	59 °F	52 °F
5:53 AM	59 °F	52 °F
6:53 AM	60 °F	52 °F
7:53 AM	63 °F	52 °F
8:53 AM	66 °F	53 °F
9:53 AM	69 °F	53 °F
10:53 AM	73 °F	54 °F
11:53 AM	75 °F	54 °F
12:53 PM	75 °F	55 °F
1:53 PM	77 °F	55 °F
2:53 PM	77 °F	55 °F
3:53 PM	76 °F	55 °F
4:53 PM	74 °F	54 °F
5:53 PM	74 °F	54 °F
6:53 PM	73 °F	54 °F

Figure 4.1: Temperature and Dew Point column

Predict Steps	MAE	RMSE	MAPE
3	3.306	5.681	6.780%
6	3.339	5.713	6.844%
9	3.652	6.339	7.616%
12	3.850	6.682	8.067%

Table 4.4: STNN with Dew Point - Closest Sensor

None of these methods showed any improvement, so we mixed up the sensor calculations and used a weighted average for softmax distances between the airport and traffic

sensors. This was also tested with wind speed in table 4.5.

Predict Steps	MAE	RMSE	MAPE
3	3.667	6.016	7.590%
6	3.633	6.045	7.584%
9	3.781	6.396	7.994%
12	3.878	6.535	8.192%

Table 4.5: STNN with Wind Speed

We would think a higher wind speed would result in drivers being more careful and slowing down around the road. However, our results prove otherwise, even worse than our models with temperature and dew point.

The condition column was a categorical variable and consisted of twenty different possible values for the PeMS region. These included values such as 'Rain/Windy', 'Fair', and 'Fog'. Several of these values appeared sparsely in the smaller dataset, such as 'Thunder in the Vicinity' appearing twice in the 1212 observations. This meant that using these values as one hot encoding would be problematic due to the small number of occurrences. Our first test was therefore run with 'Light Rain', which occurred 391 times and was one hot encoded as a new variable. Furthermore, the model hyper-parameters were tuned to weigh nearby traffic sensors with more importance. This experiment resulted in figure 4.6.

Predict Steps	MAE	RMSE	MAPE
3	2.492	4.842	5.349%
6	2.569	4.992	5.543%
9	2.682	5.187	5.803%
12	2.765	5.333	5.980%

Table 4.6: STNN with Light Rain

This showed better results overall, with all accuracy measures showing positive improvement. There were a few other conditions that were thought to be slowing down drivers that were not included as they appeared sparsely. Instead, these conditions were merged together to combine into one variable of one hot encoding. If any of the selected columns were 1, then the new column would be 1, else it would be 0. These conditions were: Rain/Windy, T-Storm, Light Rain/Windy, Rain, Thunder, Light Rain, Thunder in the Vicinity, Light Rain with Thunder, Fog, Heavy Rain, Shallow Fog, Drizzle, and Showers in the Vicinity. The results are found in table 4.7.

Again, we have a slight overall performance for this model compared to just having 'Light Rain'. This is a positive sign showing that these poor driving conditions could

Predict Steps	MAE	RMSE	MAPE
3	2.380	4.658	4.984%
6	2.443	4.803	5.130%
9	2.596	5.074	5.496%
12	2.704	5.267	5.774%

Table 4.7: STNN with Poor Conditions

cause some impact on driver speed. Because we also modified model hyper-parameters, we reran the original model for a baseline. This can be seen in table 4.8.

Predict Steps	MAE	RMSE	MAPE
3	2.327	4.578	4.824%
6	2.409	4.727	5.030%
9	2.555	4.955	5.349%
12	2.656	5.120	5.580%

Table 4.8: Original STNN

The original STNN runs better than our models with an extra feature. This could be the way how the STNN model works and how it captures the intricacies of the data to be used on other road networks. This turned to experiment with another model, the Graph Multi-Attention Network (GMAN), which focuses on a certain graph. Unfortunately, this proved problematic and was discarded to use later for another method.

Using the original input data, we manipulate the data by increasing the rate of traffic by a certain percentage due to poor weather conditions. This simulates the hypothetical scenario where the weather has not impacted traffic patterns. Experiments were run on original, 5%, 10%, 15% and a 20% increase in traffic. Prediction RMSE forecasts for 15 and 60 minutes are shown in figure 4.2 and 4.3. The following experiments are now run on the full dataset using the RTX 3090.

Predict Steps	MAE	RMSE	MAPE
3	1.199	2.421	2.499%
6	1.493	3.217	3.287%
9	1.710	3.772	3.917%
12	1.885	4.196	4.444%

Table 4.9: STNN 15%

Looking at the RMSE for 15 minute forecasts, we can see that having an increase of 15% of the original input data when the weather is considered as poor, has better RMSE accuracy compared to the original input. Even with a 10% and 20% increase, the model performs better than without this increase. From the 60-minute forecast, the original

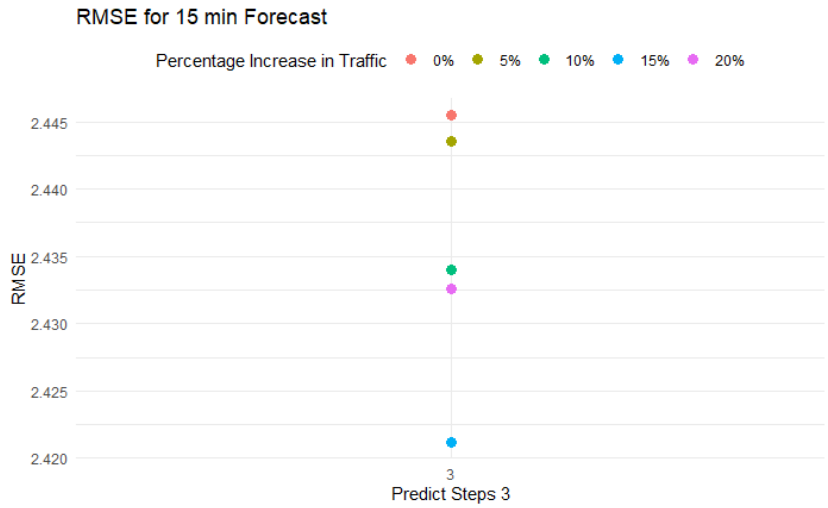

Figure 4.2: RMSE for 15 min Forecast

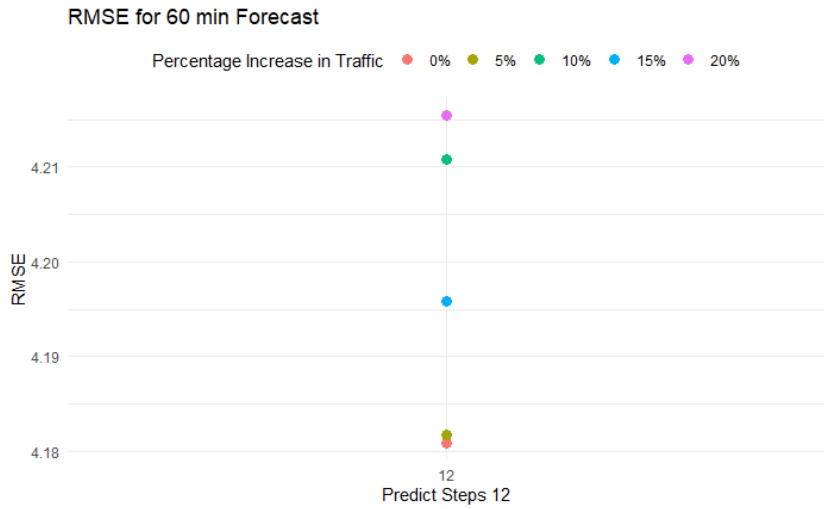

Figure 4.3: RMSE for 60 min Forecast

model performs much better, whereas modifying the input rate of traffic leads to an increased RMSE. From these, we can see that weather impacts short-term (15min) traffic rate but is poor with a longer-term (60min) forecast. We can infer from this that poor conditions are seen to impact short-term outcomes, such as heavy rain affecting traffic at that point in time, but that rain 60 minutes ago will have a less considerable effect on the drivers.

Using the GMAN model, we use the same technique of increasing the rate of traffic when poor weather conditions are met. The results can be seen in table 4.10.

The modified inputs for the GMAN model performed worse than expected, with higher RMSE values when increasing the input data. The model was scraped, and more experiments were run on the STNN model. This time, the rate is increased by 1% for each

Predict Step	Original	5% Increase	15% Increase
3	3.43	3.51	3.53%
6	4.24	4.29	4.32%
9	4.73	4.79	4.82%
12	5.24	5.23	5.22%

Table 4.10: GMAN and Modified Inputs (RMSE)

consecutive 1 up to a cap of 20%. If a 0 occurs, the rate is decreased by 2% and is also added to the data. The results are found in figure 4.11. A similar experiment was run but was increased by 2% for each consecutive 1 and is found in figure 4.12. The original STNN results are in figure 4.13 for a baseline.

Predict Steps	MAE	RMSE	MAPE
3	1.209	2.443	2.504%
6	1.497	3.247	3.293%
9	1.712	3.807	3.950%
12	1.883	4.229	4.483%

Table 4.11: 1% increase, 2% decrease

Predict Steps	MAE	RMSE	MAPE
3	1.215	2.423	2.533%
6	1.505	3.227	3.312%
9	1.718	3.783	3.949%
12	1.892	4.207	4.484%

Table 4.12: 2% increase, 2% decrease

Predict Steps	MAE	RMSE	MAPE
3	1.208	2.445	2.537%
6	1.485	3.229	3.292%
9	1.690	3.768	3.909%
12	1.857	4.181	4.425%

Table 4.13: Original STNN - Full Dataset

From our results, our 1% increase 2% decrease perform slightly worse than the original model. The 2% increase 2% decrease perform again, slightly worse. This may be due to a few factors, with the cap being set too high at 20% or the decreasing percentage being too low, which does not match normal traffic behaviour. The subsequent experiments tested more significant increase/decrease percentages along with an exponential increase and decrease. The exponential followed a 1, 2, 4, 8, 16 percent increase with consecutive ones, and a 0 decreases the percentage following the same numbers.

Predict Steps	MAE	RMSE	MAPE
3	1.212	2.460	2.534%
6	1.498	3.257	3.312%
9	1.709	3.828	3.966%
12	1.878	4.260	4.510%

Table 4.14: 5% increase, 5% decrease

Predict Steps	MAE	RMSE	MAPE
3	1.218	2.467	2.555%
6	1.495	3.253	3.303%
9	1.700	3.797	3.913%
12	1.867	4.212	4.432%

Table 4.15: 10% increase, 10% decrease

Predict Steps	MAE	RMSE	MAPE
3	1.207	2.415	2.492%
6	1.492	3.213	3.281%
9	1.705	3.771	3.926%
12	1.872	4.182	4.48%

Table 4.16: Exponential increase, Exponential decrease

The results showed neither 5% nor 10% increment values performing better than the original STNN, but the exponential experiment performed on par if not better than the original for short-term (15min) predictions. The exponential had a similar MAE but a better result for RMSE and MAPE. Comparing this to the 15% flat increase model, the exponential has a better RMSE and MAPE for short-term forecasts but a worse MAE. The exponential model was tested again but with a decrease of twice that of the increase. For example, if our increase percentage was at 16% and a 0 occurred in the next timestamp, we would increase the rate of traffic by 4% instead of the usual 8%. The new exponential had a worse accuracy value in all three measurements showing weather impacts driving speed, even when the weather is better. This could be due to wet driving conditions that have not dried up yet, or the effect of phantom traffic jams that have occurred due to poor weather.

We compare our best models of flat 15% and exponential with multiple models on the PeMS-Bay area dataset. These include Historical Averages (HA), with a moving average window size of 12 for predictions. Auto-Regressive Integrated Moving Average models augmented with a Kalman filter (ARIMA) [15]. A Recurrent Neural Network model with fully connected LSTM as hidden units (FC-LSTM) [12]. A novel method combining Diffusion Convolution and gated Recurrent Neural Network units in an encoder-decoder scheme (DCRNN) [8]. The Spatio-Temporal Graph Convolutional Networks model lever-

aging graph convolution to discern spatial patterns and a 1D convolution for temporal aspects of the graph (STGCN) [20]. Graph WaveNet utilising node embeddings to construct a dependency matrix, coupled with dilated 1D convolution for the prediction tasks (Graph WaveNet) [18]. The Graph Multi-Attention Network incorporating various attention mechanisms, including spatial, temporal and transform mechanisms to enhance forecasting (GMAN) [22]. Finally, we compare with a Multivariate Time series forecasting with Graph Convolution Networks [17]. These models have been chosen as a timeline from performing the best at traffic prediction throughout their respective years.

Models	15 mins			30 mins			60 mins		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
PeMS-Bay									
HA	2.88	5.59	6.85%	2.88	5.59	6.85%	2.88	5.59	6.85%
ARMIA	1.55	3.01	3.12%	1.98	4.12	4.76%	2.71	5.34	6.62%
FC-LSTM	1.89	3.88	4.40%	2.01	4.19	4.75%	2.16	4.38	5.24%
DCRNN	1.28	2.63	2.57%	1.51	3.49	3.40%	1.78	4.19	4.28%
STGCN	1.27	2.65	2.60%	1.58	3.88	3.70%	2.20	4.63	4.87%
Graph WaveNet	1.22	2.45	2.50%	**1.47**	3.31	3.21%	1.68	3.56	3.88%
GMAN	1.26	2.50	2.58%	**1.47**	3.32	**3.18**%	**1.65**	**3.53**	**3.61**%
MTGCN	1.24	2.46	2.53%	1.50	3.34	3.30%	1.69	3.65	3.90%
Original STNN	1.2083	2.4455	2.5366%	1.4849	3.2288	3.2923%	1.8568	4.1808	4.4254%
15% STNN	**1.1990**	2.4212	2.4992%	1.4930	3.2171	3.2869%	1.8850	4.1958	4.4437%
Exponential STNN	1.2070	**2.4151**	**2.4923**%	1.4924	**3.2127**	3.2809%	1.8721	4.1815	4.4482%

Table 4.17: Overall performance of short-term (15 mins), mid-term (30 mins) and long-term (60 mins) traffic forecasting

From the overall results, we can see that by increasing the traffic rate of input data by 15% when we see poor driving conditions, we can improve the MAE for short-term traffic prediction. If using an exponential increase capping at 16%, we have a better performance in RMSE and MAPE. The mean absolute error measures the average magnitude of errors regardless of the direction. RMSE squares the difference before averaging, giving greater weight to larger errors. One possible scenario could be if the 15% STNN makes many small errors and some large ones. In contrast, the Exponential STNN makes slightly greater errors on average, as indicated by the slightly higher MAE, but it makes fewer large errors, as indicated by the lower RMSE. Similarly, the lower MAPE of the Exponential STNN model means that its mistakes are a smaller fraction of the actual values it predicts, implying that it may perform better on higher traffic flow numbers, such as the highway in the PeMS-Bay area data.

This research sheds light on how including weather data as an extra feature might increase the accuracy of traffic forecast algorithms. Weather is just one of these many variables, but our data indicate that it may impact traffic patterns.

5
Conclusion

In this paper, we incorporated meteorological data in surrounding traffic networks to improve the accuracy for rate of traffic. We took weather data from Wunderground and captured specific conditions where drivers would be impacted by the environment. We compared this with the baseline of state-of-the-art traffic graph neural networks and our model showed improvement in short to medium term forecasting. We first proposed to include weather conditions as an additional variable which saw no significant improvement. Then, we tried reversing the traffic flow caused by poor weather which was seen effective. We improved on all three accuracy measures MAE, RMSE and MAPE on the 15-minute forecast as well as RMSE on the 30-minute forecast.

Suggestions for Future Work

There is another solution instead of manipulating the input data. The traffic speed as input data can be transformed into a d-dimensional embedding vector, 'v', by multiplying it with a $1 \times d$ matrix, 'W_1'. The binary weather information can also be encoded into two d-dimensional embedding vectors, 'v_1' representing poor weather where driving conditions will be affected and 'v_0' meaning clear weather where driving conditions are normal. For each 5-minute time step, if the weather is poor, the input to the STNN model would be '$v + v_1$'; otherwise, it would be '$v + v_0$'. These parameters (W_1, v_1, v_0) could be optimized alongside the STNN model, making them adaptable to the weather data. This proposed approach is theoretically better than manipulating the original data. This is because it aligns the two types of features, traffic speed and weather conditions, into the same feature space via a linear projection. This method may better capture the underlying interactions between traffic speed and weather conditions and could potentially enhance the accuracy of the STNN model's predictions.

Furthermore, there are many personal weather stations on Wunderground, these include daily readings similar to airports but the website does not hold any history data. It is possible to use this for more weather sensors while training the model but may be prone to overfitting as weather does not change heavily with distance. We have also found out that some cities have very little rain as found out in METR-LA data. We can push this model onto datasets where the weather rains more and where the conditions of the road are less favourable compared to a normal sunny day. These could be San Diego and Minneapolis Minnesota Twin City datasets where the weather fluctuates much more greater with more rain compared to PeMS-Bay area.

Bibliography

[1] Kit Yan Chan, Tharam S Dillon, Jaipal Singh, and Elizabeth Chang. Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and levenberg–marquardt algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):644–654, 2011.

[2] Rong-Sheng Chen, Chung-Yi Liang, Wei-Chiang Hong, and Ding-Xuan Gu. Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm. *Applied Soft Computing*, 26:435–443, 2015.

[3] Guowen Dai, Changxi Ma, and Xuecai Xu. Short-term traffic flow prediction method for urban road sections based on space–time analysis and gru. *IEEE Access*, 7:143025–143035, 2019.

[4] Stephen Dunne and Bidisha Ghosh. Weather adaptive traffic prediction using neurowavelet models. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):370–379, 2013.

[5] United States Department of Transportation Federal Highway Administration. Average annual miles per driver by age group, May 2022.

[6] Yue Hou, Zhiyuan Deng, and Hanke Cui. Short-term traffic flow prediction with weather conditions: based on deep learning algorithms and data fusion. *Complexity*, 2021:1–14, 2021.

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

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

[9] Marco Lippi, Matteo Bertini, and Paolo Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882, 2013.

[10] Archana Nigam and Sanjay Srivastava. Macroscopic traffic stream variables prediction with weather impact using hybrid cnn-lstm model. In *Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking*, pages 1–6, 2021.

[11] Iwao Okutani and Yorgos J Stephanedes. Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):1–11, 1984.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[13] Yongxue Tian and Li Pan. Predicting short-term traffic flow by long short-term memory recurrent neural network. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, pages 153–158. IEEE, 2015.

[14] Billy M Williams, Priya K Durvasula, and Donald E Brown. Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transportation Research Record*, 1644(1):132–141, 1998.

[15] Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.

[16] Bob Williams. *Intelligent transport systems standards*. Artech House, 2008.

[17] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.

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

[19] Song Yang, Jiamou Liu, and Kaiqi Zhao. Space meets time: Local spacetime neural network for traffic flow forecasting. *arXiv preprint arXiv:2109.05225*, 2021.

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

[21] Jiandong Zhao, Yuan Gao, Yunchao Qu, Haodong Yin, Yiming Liu, and Huijun Sun. Travel time prediction: Based on gated recurrent unit method and data fusion. *IEEE Access*, 6:70463–70472, 2018.

[22] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1234–1241, 2020.

[23] Eric Zivot and Jiahui Wang. Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®*, pages 385–429, 2006.