UNIVERSITY OF CALIFORNIA,
IRVINE

Imputation of Missing Traffic Flow Data by Using Denoising Autoencoders

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Civil and Environmental Engineering

by

Boyuan Jiang

Dissertation Committee:
Professor R. (Jay) Jayakrishnan, Co-Chair
Professor Amelia Regan, Co-Chair
Assistant Professor Michael Hyland

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to two committee chairs, Professor R. (Jay) Jayakrishnan and Professor Amelia Regan, who helped me a lot during the whole process of research and thesis writing. Without their guidance and persistent help, this thesis would not have been possible.

I would like to thank Professor Michael Hyland of the Civil and Environmental Department and two Computer Science Department professors: Michael Dillencourt and Kalev Kask who gave me helpful discussions about this research topic.

Besides, a thank you to Muhammad Danial Siddiqi, who worked with me since January, and Reza Asadi who introduced this research direction and gave me a lot of suggestions these several months.

I thank my parents and all my friends for supporting me during this hard pandemics time.

ABSTRACT OF THE THESIS

Imputation of Missing Traffic Flow Data by Using Denoising Autoencoders

by

Boyuan Jiang

Master of Science in Civil and Environmental Engineering

University of California, Irvine, 2020

Professor R. (Jay) Jayakrishnan, Co-Chair

Professor Amelia Regan, Co-Chair

In transportation engineering, Spatio-temporal data including traffic flow, speed, and occupancy are collected from different kinds of sensors and used by transportation engineers for analysis. However, the missing data influence the analysis and prediction results significantly. In this thesis, Denoising Autoencoders are used to impute the missing traffic flow data. First, we focused on the general situation and used three kinds of Denoising Autoencoders: “Vanilla”, CNN, and Bi-LSTM to implement the data with a general missing rate of 30%. Each model was optimized by focusing on the main hyper-parameters since the tuning can influence the accuracy of the final prediction result. Then, the Autoencoder models are used to train and test data with an exceptionally high missing rate of about 80%. We do this to test and then demonstrate that even under extreme loss conditions, Autoencoder models are very robust. By observing the hyper-parameter tuning process, the changing prediction accuracy is shown and in most cases, all three models maintain good accuracy even under the worst situations. Moreover, the error patterns and trends concerning different sensor stations and different hours on weekdays and weekends are also

visualized and analyzed. Finally, based on these results, we separate the data into weekdays and weekends, train and test the models respectively, and improve the accuracy of the imputation result significantly.

CHAPTER 1: Introduction

Spatio (Spatial) refers to space. Temporal refers to time. Spatio-temporal or spatial-temporal is used while data are collected over both space and time [1]. Spatio-temporal problems where the data are collected by a great number of sensors over a long period have been studied in transportation engineering decades. However, due to the sensor malfunctions, communication failure, or measurement errors [2], part of the data will be inevitably be missed. Not only will the presence of missing data influence the final result of traditional transportation data analysis, but also the performance of the machine learning models will drop significantly on the tasks of clustering and prediction. To overcome this challenge, data must be processed and the imputation of missing data is one way that improving the accuracy of final results.

For missing data imputation, statistical and machine learning techniques and methods are broadly used. A widely used and popular statistical method for time series forecasting is the ARIMA model which is short for **Auto Regressive Integrated Moving Average** [3]. The parameters that must be determined for an ARIMA model include the level of autoregression (AR), the level of integration (I), and the moving average (MA). ARIMA is a class of models that 'explains' a given time series based on its past values which are its lags and the lagged forecast errors, so that equation can be used to forecast future values. Since Spatio-temporal data contains too many features for any model to be able to train with a reasonable amount of computation, dimensionality reduction techniques are introduced. One such dimensionality reduction technique is probabilistic principal component analysis, where **Principal Component Analysis** (**PCA**) can be used to extract features of traffic flow

[4]. PCA is a technique for feature extraction so it combines the various input variables in a specific way, then the “least important” variables can be dropped while the most valuable parts of all of the variables are still retained. As Spatio-temporal datasets become larger, it’s hard to extract features only by using handcrafted feature engineering. Instead, deep learning can be used where the model can search and find the dataset feature by themselves without requiring any assumptions.

The utilization of deep learning modeling techniques such as denoising autoencoders has been shown to dominate traditional statistical and machine learning models [5]. Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. Autoencoders consist of three main components: encoder, bottleneck, and decoder. **Encoder** helps the model to learn how to reduce the input dimensions and compress the input data into an encoded representation. The **bottleneck** is the layer that contains the compressed representation of the input data. This is the lowest possible dimension of the input data. And the **decoder** is the part of the model that learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.

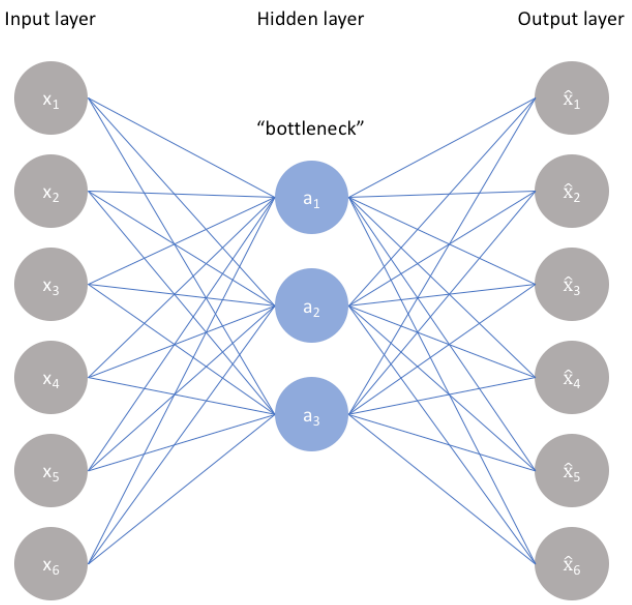

Figure 1 Process of encoder and decoder

An autoencoder's purpose is to map high-dimensional data (e.g images) to a compressed form (i.e. hidden representation) and build up the original image from the hidden representation. However, when there are more nodes in the hidden layer of the Autoencoder than there are inputs, the Network is at risk of learning the so-called "Identity Function", also called the "Null Function", meaning that the output equals the input, rendering the Autoencoder useless. **Denoising Autoencoders** solve this problem by corrupting the data on purpose by randomly turning some of the input values to zero. In general, the percentage of input nodes that are being set to zero is about 50%. Other sources suggest a lower count, such as 30%. It depends on the amount of data and the input nodes. When calculating the Loss function, it is important to compare the output values with the original input, not with the corrupted input. That way, the risk of learning the identity function instead of extracting features is eliminated. A denoising autoencoder, in addition to learning to compress data (like an autoencoder), learns to remove noise in images, which

allows it to perform well even when the inputs are noisy. So denoising autoencoders are more robust than autoencoders and they learn more features from the data than a standard autoencoder.

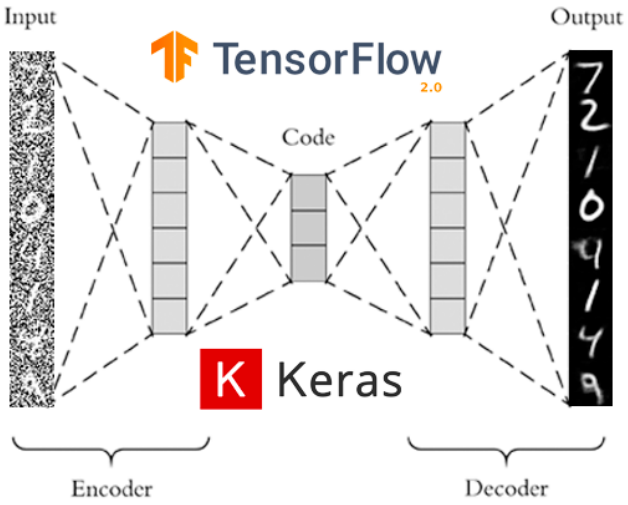

Figure 2 Example of Autoencoder

Recent advances in deep learning models provide more capability in exploring Spatio-temporal problems [6]. Denoising autoencoders are the most common deep learning models used for missing data imputation. We consider missing data imputation in traffic flow data, which is complex Spatio-temporal data, using denoising autoencoders. In [7], they examine the performance of denoising autoencoders for missing data imputation, however, their analysis is not for Spatio-temporal data. In [8], a layer-wise pre-training of fully connected layers is proposed to impute missing traffic flow data. In [2], they use denoising stacked autoencoders for missing traffic flow data imputation. However, they only consider fully connected layers, and they did not consider convolutional and recurrent neural networks. Moreover, multiple imputations of denoising autoencoders are examined in [9]. In [10], they consider a deep convolutional neural network for missing data imputation. In [11], they show the outperformance of recurrent neural networks for missing data imputation in time-

series data, and in [12], a convolutional-recurrent neural network is proposed for missing data imputation of Spatio-temporal data. Many of these works focus on proposing fully-connected, convolutional, and recurrent denoising autoencoders with better performance for missing data imputation. In this paper, we examine the performance of fully connected, convolutional, and bi-directional LSTM denoising autoencoders for missing traffic flow imputation. We explore various missing data ratios. We also explore the variation of missing data imputation error for spatial and temporal contexts, as missing data imputation error varies over the spatial and temporal domain. Such an analysis illustrates the capability of implemented autoencoders in imputing missing data under various missing data ratios and spatial and temporal domains.

We focused on the general situation and used three kinds of Denoising Autoencoders to implement the data with a general missing rate of 30% and a high missing rate of 80%, verified the robustness of Autoencoder models, concluded the error patterns and trends, and also improved the accuracy of the imputation result.

CHAPTER 2: Preliminaries

2.1 Dataset

The traffic flow data extracted from the PeMS (Caltrans Performance Measurement System) dataset are used in this study. In the original dataset, traffic flow data are gathered every 30 seconds and aggregated every 5 minutes using the loop detectors, magnetometers, and radars. We chose the region 1 stations in the Bay Area, as shown in Figure 3, to implement our models. This region has 26 mainline stations on 22 miles of highway US 101-South and the middle 8 stations data are used in this research. The selected sensors have more than 99% of the available data for this period. Part of traffic flow data are shown in Figure 4.

Figure 3 Distribution of Stations in the Bay Area

	Flow.94	Flow.95	Flow.97	Flow.100	Flow.101	Flow.104	Flow.106	Flow.107
2016-01-01 00:00:00	51.0	52.0	26.0	38.0	52.0	63.0	67.0	88.0
2016-01-01 00:05:00	52.0	55.0	21.0	34.0	52.0	57.0	60.0	88.0
2016-01-01 00:10:00	40.0	36.0	17.0	35.0	60.0	66.0	80.0	105.0
2016-01-01 00:15:00	34.0	40.0	34.0	61.0	104.0	103.0	104.0	176.0
2016-01-01 00:20:00	68.0	79.0	48.0	72.0	142.0	150.0	156.0	156.0
...	...	...	...	...	...	...	...	...
2016-04-28 23:35:00	91.0	101.0	60.0	87.0	98.0	116.0	129.0	191.0
2016-04-28 23:40:00	97.0	95.0	58.0	79.0	104.0	119.0	147.0	204.0
2016-04-28 23:45:00	101.0	91.0	72.0	75.0	116.0	125.0	145.0	189.0
2016-04-28 23:50:00	96.0	93.0	57.0	76.0	110.0	125.0	139.0	186.0
2016-04-28 23:55:00	94.0	103.0	69.0	85.0	107.0	96.0	132.0	173.0

Figure 4 Sample of Data

2.2 Problem definition

Spatio-temporal data is represented by a matrix $X \in \mathbb{R}^{s \times t \times f}$, where s is the number of sensors, t is the number of time steps and f is the number of features. In this research, only traffic flow data and 8 stations are considered, so the s is 8 and f is 1, and the matrix is $\in \mathbb{R}^{8 \times t \times 1}$. In the machine learning prediction problem, the error between the true value and prediction value is used to judge the accuracy of one model so that we need to do pre-processing work to our raw dataset. First, the missing part of the original dataset is filled using adjacent values, and a “perfect” dataset is formed. Then, due to the fact that missing data may exist at individual points or for some time periods, the random parts of the dataset are processed to be missed artificially at random length, and the missing dataset is formed. Finally, the training and testing process is implemented, and RMSE and MAE are used to calculate the accuracy of each model.

2.3 Sliding window

To apply the neural network model for time series imputation, a sliding window is needed to generate the data point for each time stamp and the parameter "look back" is used to adjust the shape of each data point. The dimension of each data point is $1 \times (s \times f \times l)$ where l is the value of look back. In our case, this can be simplified to $1 \times (8 \times 1 \times l)$ due to the number of sensors and features that have been fixed to 8 and 1. For example, if we consider the previous 4-time stamps for each data point that means the l is 4, each data point will be the matrix $\mathbb{R}^{1\times32}$.

2.4 Models

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. An autoencoder aims to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". The autoencoder can be used in the missing data imputation problem. In the training process, the model will take a matrix with missing data X^i_m as input and a "perfect" matrix X^i as a target. Autoencoders learn some latent representation which is a term for hidden features of the data and use that to reconstruct the data.

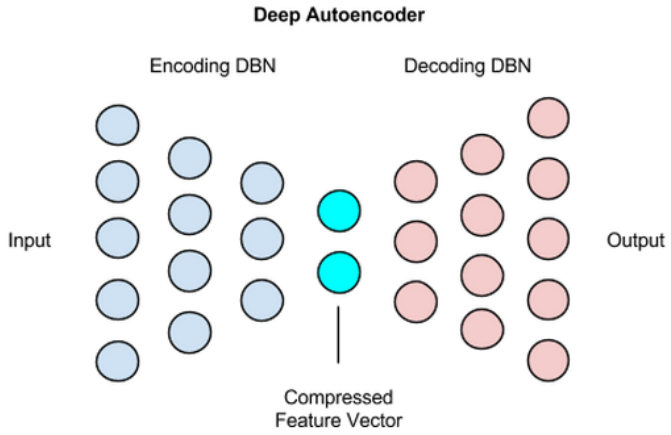

Figure 5 Autoencoder

We will consider 4 different autoencoder methods for imputation of missing data. The first is used as an industry-wide accepted method to predict traffic flow. The next 3 are various deep learning model implementations of Denoising Autoencoders.

a) Weekly-Hourly Average

Our first missing data imputation method uses a temporal average to fill missing data. Traffic flow patterns are repeated every week. Hence, a weekly-hourly average table is obtained from training data. The main drawback of using the temporal average is that specific days such as holidays or event days (games, festivals, concerts) have their patterns and they are not repeated in the training data. This is the industry-wide accepted method to predict traffic flow.

b) Vanilla (FCN)

This is our baseline deep learning model. It is a relatively simple machine learning model with only one hidden layer to create a Fully Connected Network (FCN).

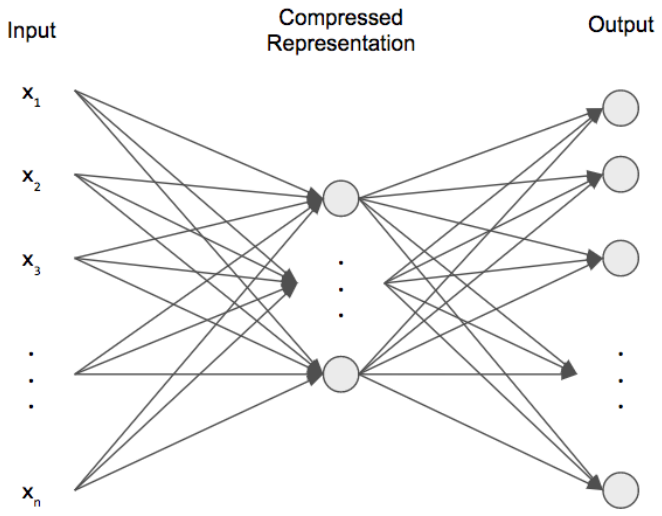

Figure 6 Fully Connected Network

c) CNN

A Convolutional Neural Network (CNN) uses convolutional layers with interim pooling layers (max) between each convolutional. We will pass our data from the sliding window as is so the model can process the 2D representation of the input data.

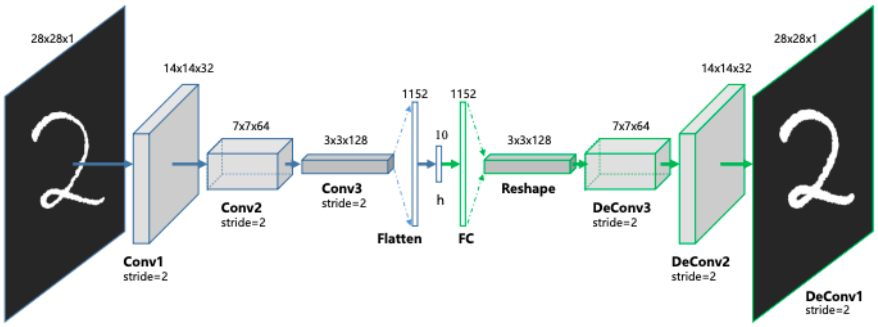

Figure 7 CNN Autoencoder

d) Bi-LSTM

We will use bidirectional Long Term Short Term Memory (LSTM) layers to take into account the previous and future values of the model to increase the model's accuracy.

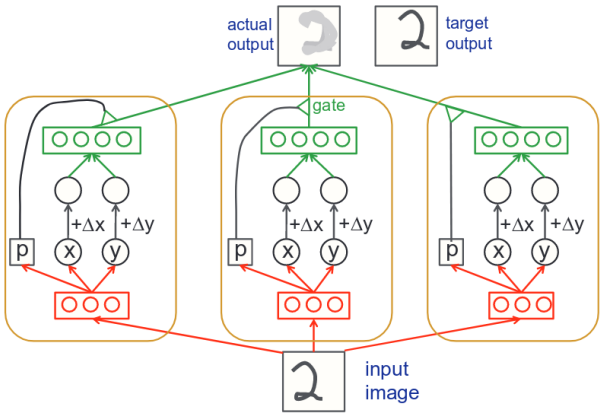

Figure 8 Bi-LSTM Autoencoder

2.5 Hyperparameters

The hyper-parameters of tuning are shown as follows:

Table 1 Hyper-parameters of tuning

Look Back	This represents the time interval of the sliding window.
Batch Size	This represents the number of samples that are processed before the model is updated.
Epochs	This represents the number of times that the dataset is used for training the model.
Units	This represents the number of units in the hidden or LSTM layer used for feature extraction.
Structure	This represents the CNN structure: layers and units used for CNN.

2.6 Comparison metrics

We will look at 5 measurements to compare the results of tuning the hyperparameters. For each of the measurements, a lower value is better.

Table 2 5 measurements for comparison

Training Loss	This represents the loss measured on the training set.
Validation Loss	This represents the loss measured on the validation set.
RMSE	This represents the Root Mean S8quare Error of the predicted missing values only.
MAE	This represents the Mean Absolute Error of the predicted missing values only.
Epoch Time	This represents the time taken for one Epoch.

To compare the results, we will rank the measurements in the following order.

Table 3 Rank of measurements

RMSE	Gives high weight to large errors, so it is much more sensitive to wrong predictions.
MAE	Gives a general overview of the error rate.
Validation Loss	Indicates overfitting or underfitting when compared with Training Loss.
Epoch Time	Indicates the cost of computation.

CHAPTER 3: Experimental results for low missing rate

The missing blocks to our training and testing dataset are added manually (the original data are complete) and the model's output is compared to the original data without the missing blocks. The missing blocks are represented by the value -1. In the section with a low missing rate, about 28.9% of the training dataset is missed for training and about 14.5% of the dataset is missed for testing. A sample of the training set with the missing part, complete training set, testing set with missing part, and complete testing set are shown in Figures 9 to 12.

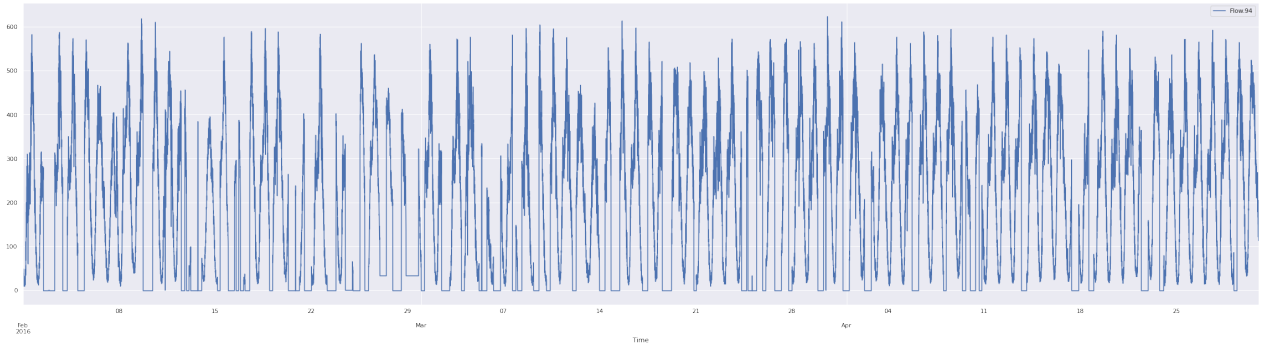

Figure 9 Training dataset with about 28.9% missing rate (low)

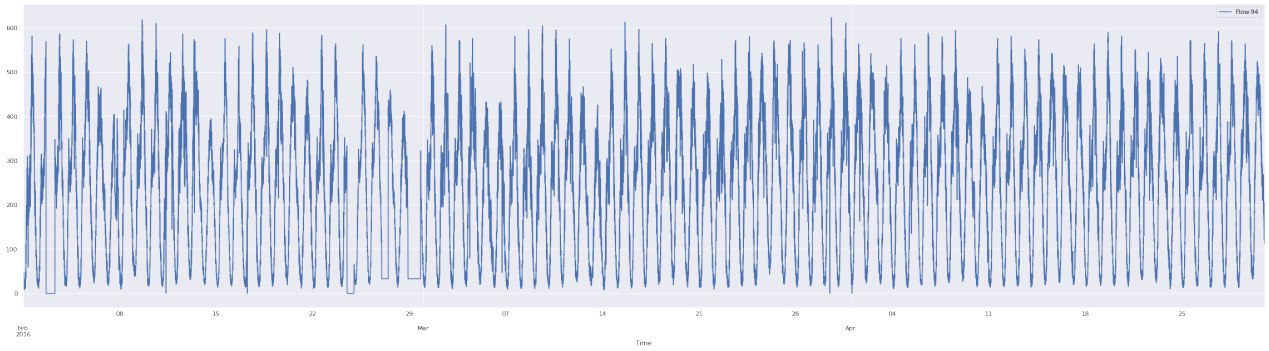

Figure 10 Complete training dataset

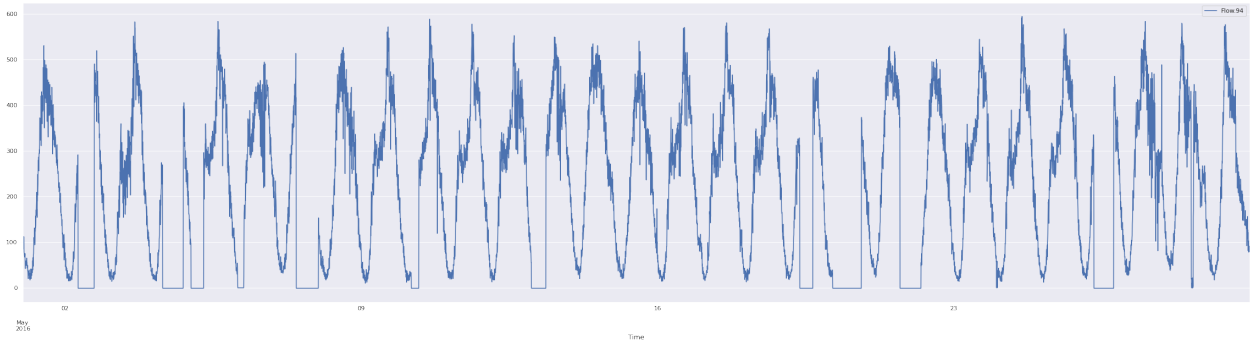

Figure 11 Testing dataset with about 14.5% missing rate

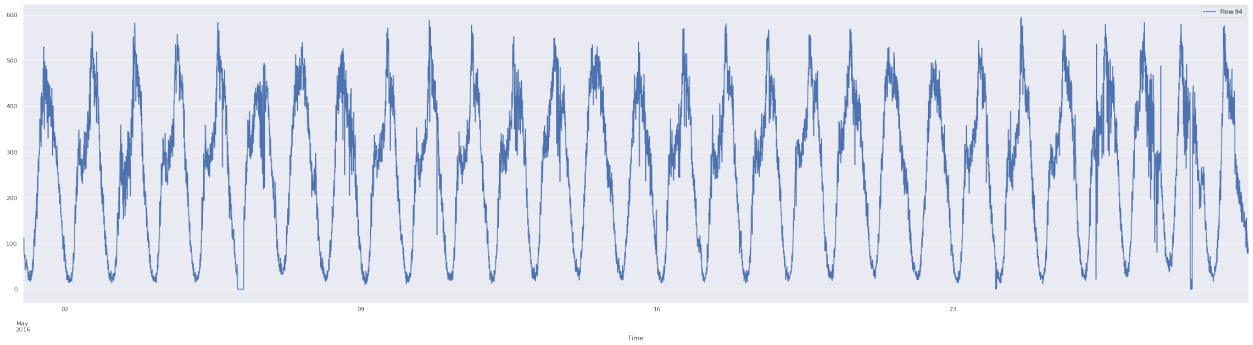

Figure 12 Complete testing dataset

In this section, three models: “Vanilla”, CNN, and Bi-LSTM are considered and their hyper-parameter tuning results are shown as follows.

3.1 Effects of hyperparameter tuning on “Vanilla” (FCN)

For “Vanilla”, four hyper-parameters which are “look back”, “batch size”, “epochs”, and “units” are considered and tuned.

Look Back (time interval of the sliding window)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Look Back	4	48.4	18.43	0.3	0.0009	0.0053
	8	47	17.59	0.3	0.0007	0.0051
	12	47.5	19.39	0.4	0.0009	0.005
	36	54.9	26.75	0.9	0.0014	0.0066

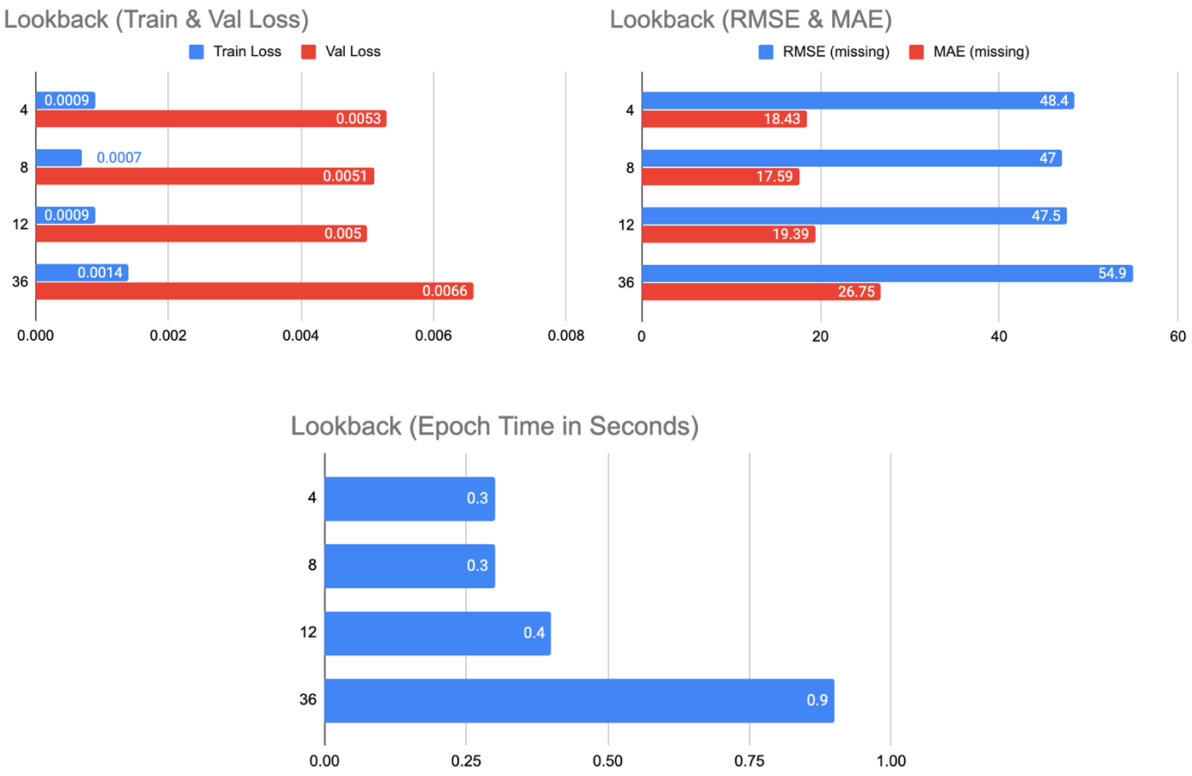

Figure 13 Tuning of look back for "Vanilla"

As the increase of look back, both training and validation loss fall at first and then rise, and the same for RMSE and MAE. The epoch time increases continuously. The optimal lookback value seems to be 8 since it provides the smallest RMSE (47) and MAE (17.59) values. Validation Loss for a look back of 8 (0.005) is the same as that of 12 (0.0051) but the computational cost is lower (0.3).

Batch Size (number of samples that are processed before the model is updated)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Batch Size	256	47.5	18.24	0.3	0.0009	0.0055
	512	32.9	16.31	0.3	0.0022	0.0028
	1024	33.3	17.33	0.3	0.0025	0.0028

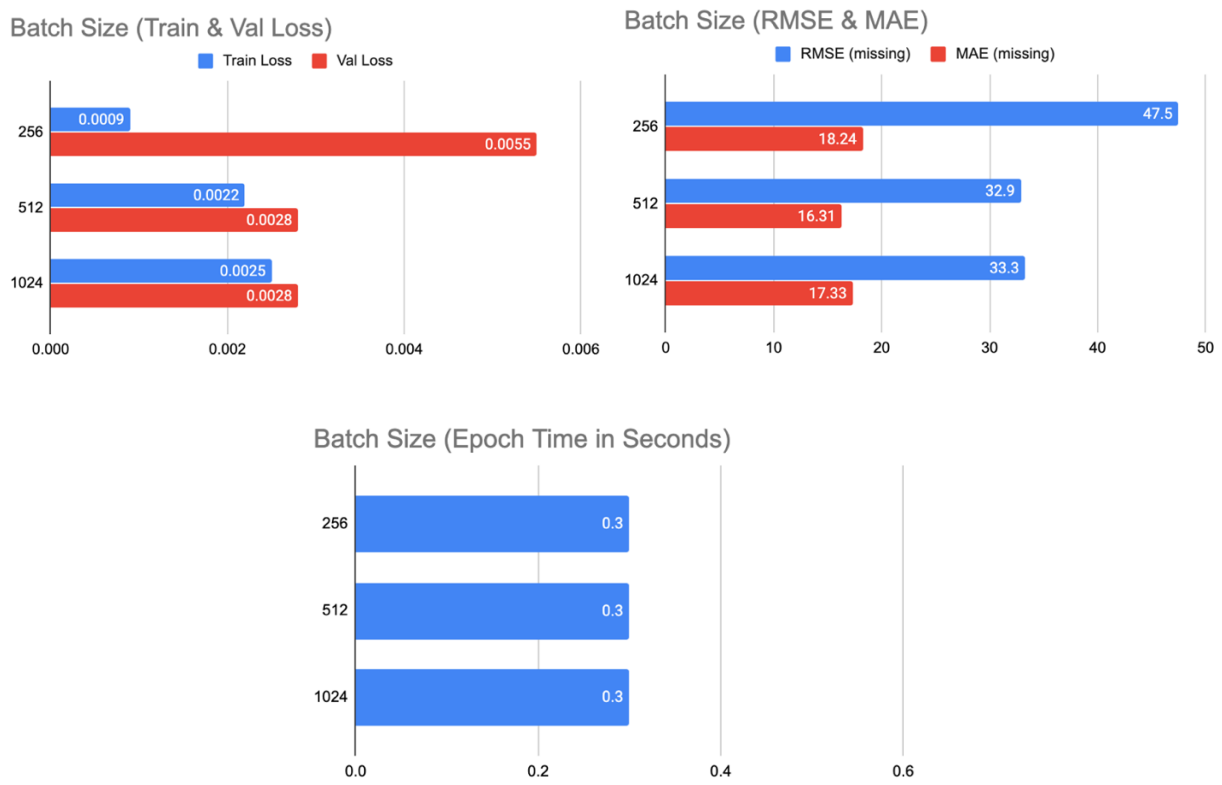

Figure 14 Tuning of batch size for "Vanilla"

With the increase of batch size, the training loss increases while the validation loss decrease, both RMSE and MAE first fall then rise, and the epoch time remain unchanged. The optimal batch size value seems to be 512 since it provides the smallest RMSE (32.9) and MAE (16.31) values. The batch size value of 256 has a lower Training Loss (0.0009) than 512 (0.0022), but the Validation Loss (0.0055) is much higher. This seems to indicate overfitting.

Epochs (number of the times that the dataset is used for training the model)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Epoch	50	35.3	19.06	0.3	0.003	0.0032
	100	33.1	17.41	0.3	0.0026	0.0028
	200	31.7	15.54	0.3	0.0022	0.0026
	500	37.5	12.39	0.3	0.0013	0.0038

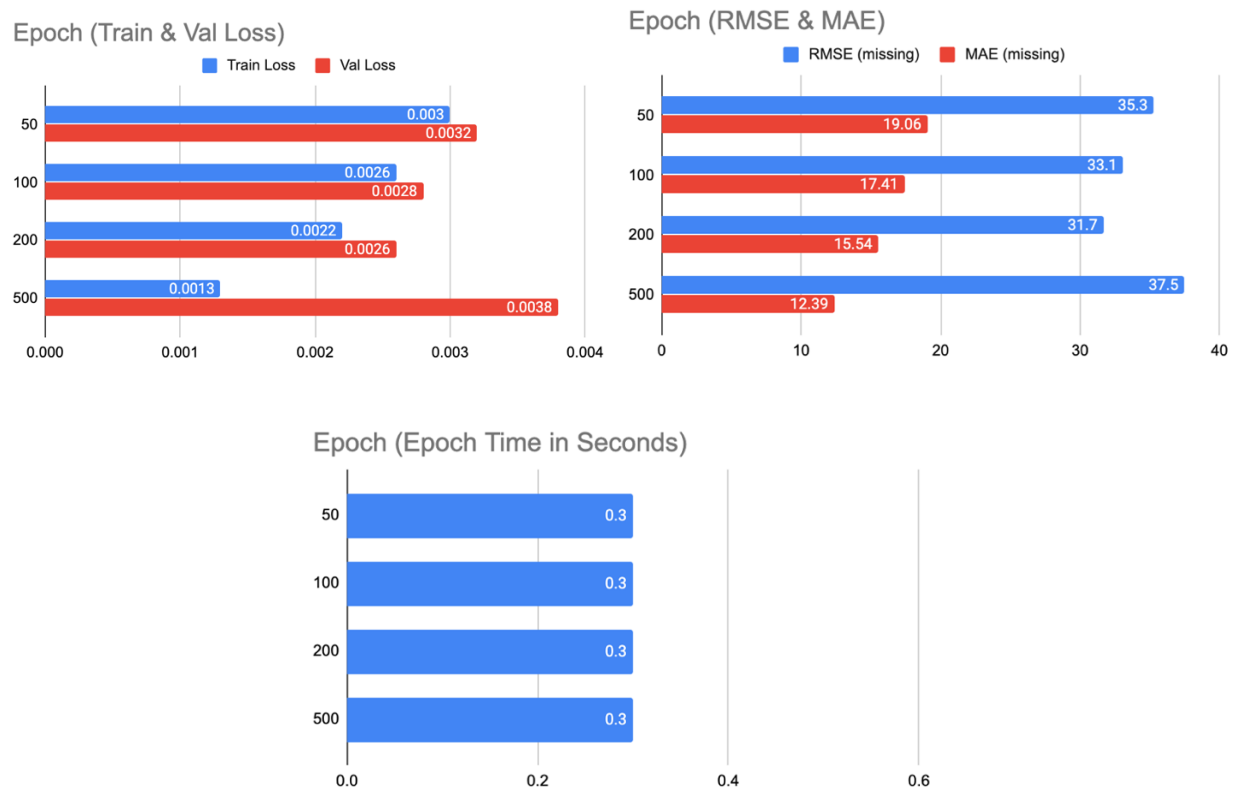

Figure 15 Tuning of epochs for "Vanilla"

As the increase of epoch, the training loss drops continuously while the validation loss first drops then rises, the MAE drop continuously while the RMSE first drop then rises, and the epoch time is all the same. The optimal epoch value seems to be 200 since it provides the smallest RMSE (31.7). While the MAE value is better for 500 (12.39), the RMSE value of 500 (37.5) is worse which implies that the margin of error is higher. Higher validation loss for 500 (0.0038) seems to indicate overfitting hence the higher margin of error.

Units (number of units in the hidden layer used for feature extraction)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Units	64	33.6	15.88	0.3	0.0021	0.0029
	96	32.5	14.27	0.3	0.0019	0.0027
	144	30.6	12.47	0.3	0.0015	0.0025
	256	31.8	10.27	0.5	0.0011	0.0027
	512	29.2	9.06	0.7	0.0009	0.0022
	1024	33.3	9.11	1.4	0.0007	0.0029

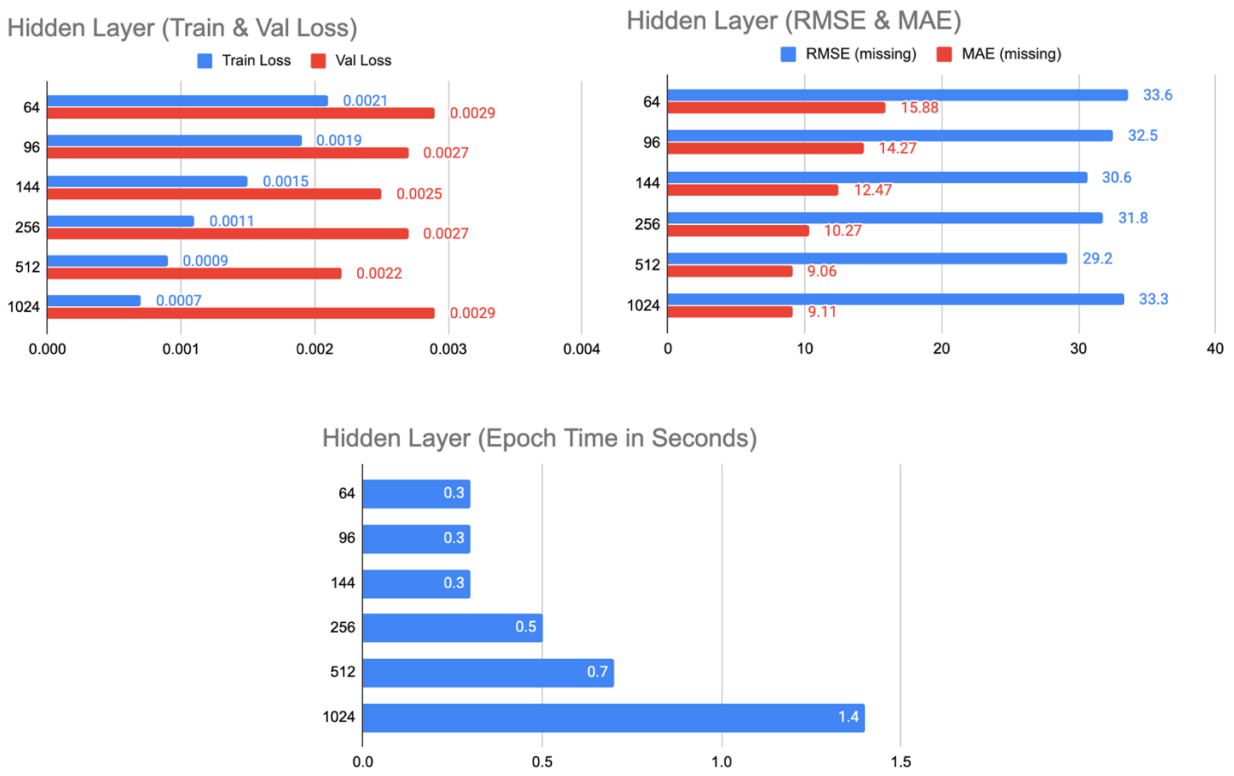

Figure 16 Tuning of units for "Vanilla"

As the increase of epoch, the training loss drops continuously while the validation loss fluctuates near 0.0025, the RMSE also fluctuates while the MAE almost drops continuously, and the epoch time rises continuously. The optimal unit value seems to be 512 since it provides the smallest RMSE (29.2). While the MAE value is similar to 1024 (9.11), the RMSE value of 1024 (33.3) is worse which implies that the margin of error is higher. Higher validation loss for 1024 (0.0029) seems to indicate overfitting hence the higher margin of error. 1024 has a much higher computation cost (1.4) as well.

3.2 Effects of hyperparameter tuning on CNN

For CNN, three hyper-parameters which are "look back", "batch size", and "epochs" are considered and tuned.

Look Back (time interval of the sliding window)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Look Back	4	42.2	21.03	3	0.5377	0.549
	8	36.1	19.75	5	0.5385	0.5467
	12	36.4	20.2	6	0.538	0.5464
	36	36.7	20.03	17	0.5378	0.5467

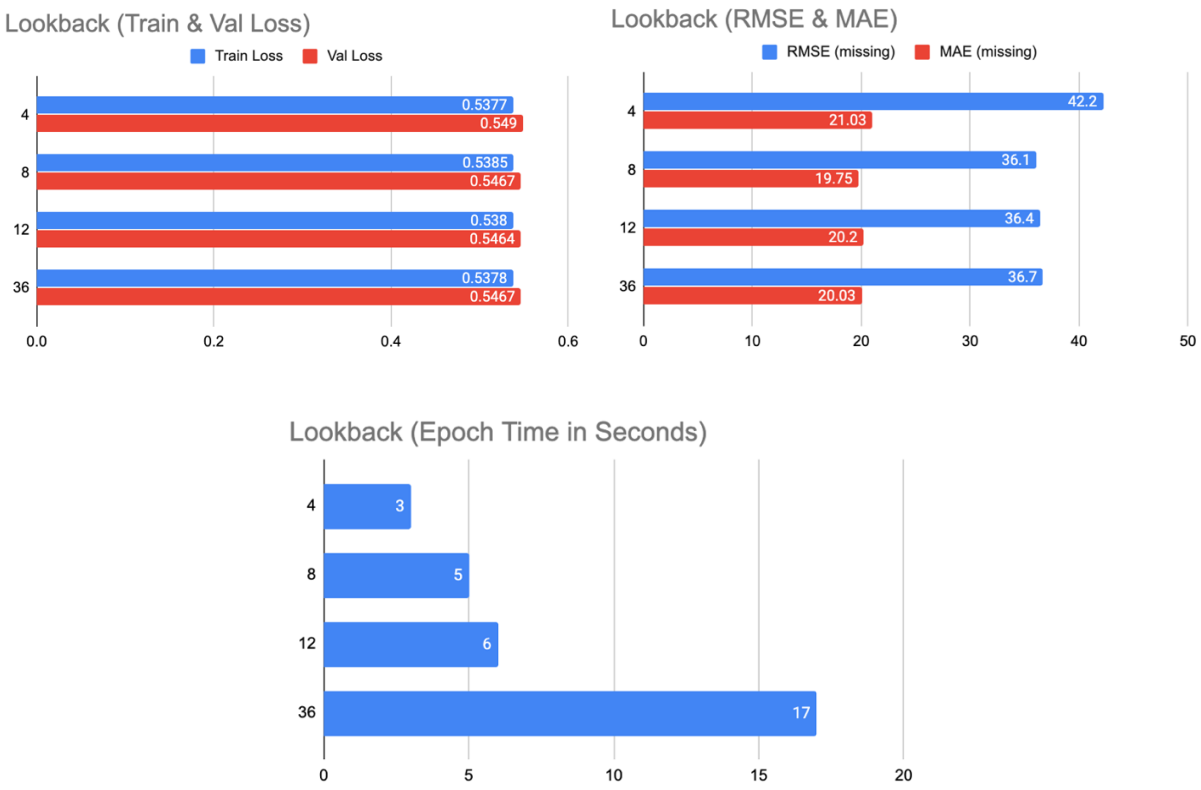

Figure 17 Tuning of look back for CNN

As the increase of look back, the training loss and validation loss are almost the same, the RMSE first drop then rise slowly while the MAE is also almost the same, and the epoch time rises continuously. The optimal lock back value seems to be 8 since it provides a similar RMSE (36.1) and MAE (19.75) values to 12 (36.4, 20.2) and 36 (36.7, 20.03) but the computational cost is lower (5).

Batch Size (number of samples that are processed before the model is updated)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Batch Size	128	38.9	19.13	6	0.5384	0.5472
	256	36.1	19.75	5	0.5385	0.5467
	512	39.12	21.71	4	0.5392	0.5475

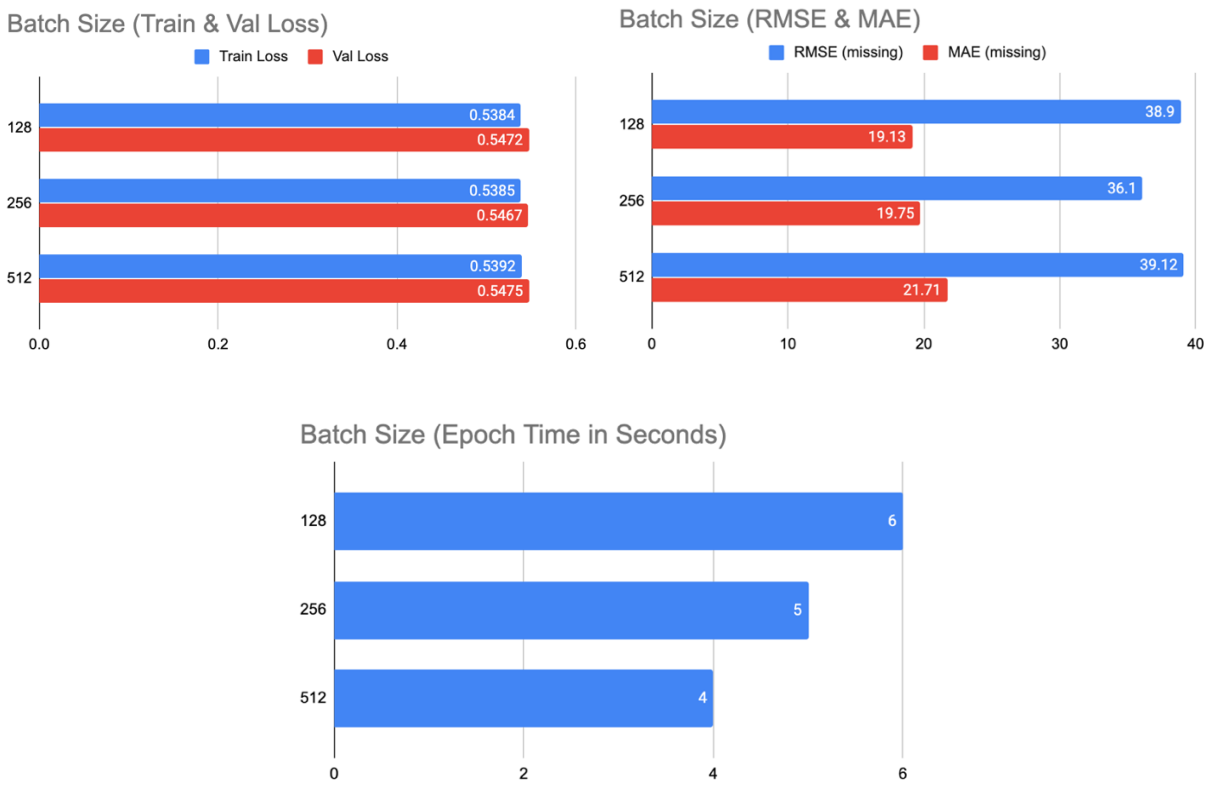

Figure 18 Tuning of batch size for CNN

As the increase of batch size, the training loss and validation loss are almost the same, the RMSE first drop then rise while the MAE rise continuously, and the epoch time falls continuously. The optimal batch size value seems to be 256 since it provides the smallest RMSE (36.1), and MAE (19.75) value which is similar to 128 (19.13), but the cost of computation for a batch size of 256 (5) is lower than that of 128 (6).

Epochs (number of the times that the dataset is used for training the model)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Epoch	50	36.7	22.71	5	0.5394	0.5474
	100	36.1	19.75	5	0.5385	0.5467
	200	32.8	17.11	5	0.5376	0.5453
	500	34.9	18.11	5	0.5364	0.5465

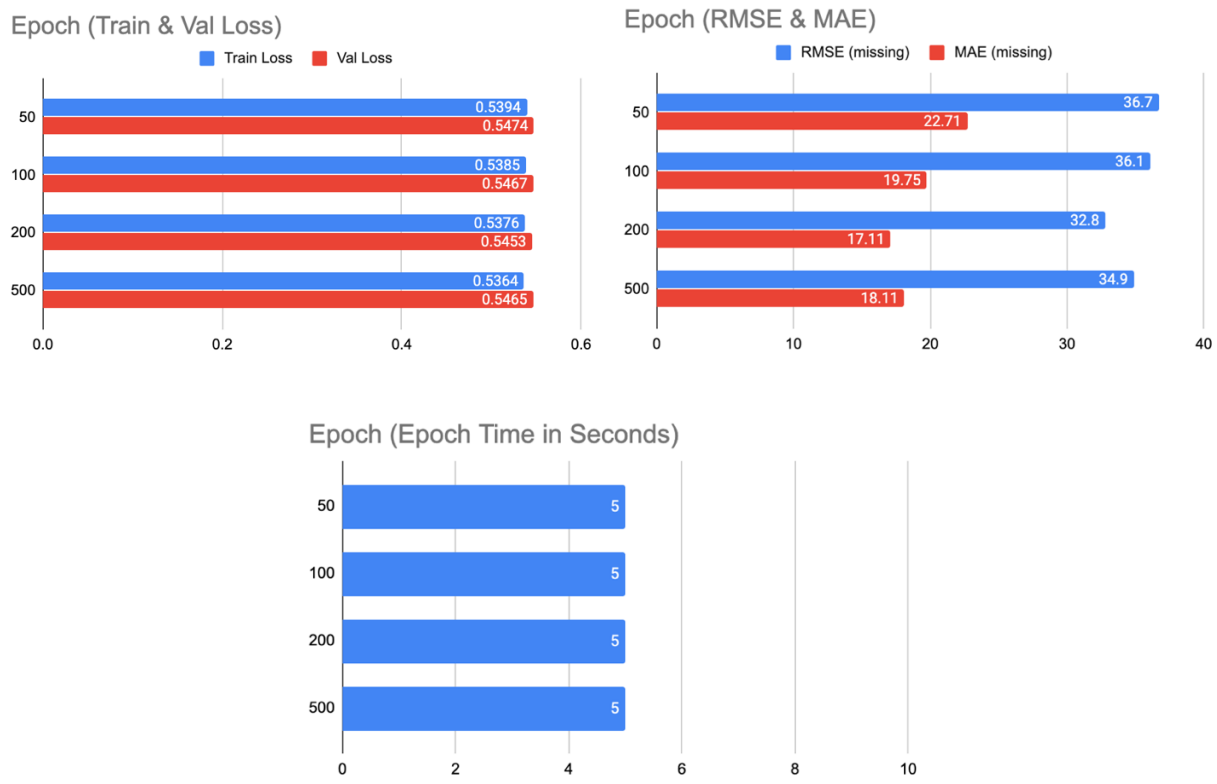

Figure 19 Tuning of epochs for CNN

As the increase of epochs, the training loss and validation loss are still almost the same, the RMSE first falls then rises and this trend is the same for MAE, and the epoch time is almost the same. The optimal epoch value seems to be 200 since it provides the smallest RMSE (32.8) and MAE (17.11) values.

3.3 Effects of hyperparameter tuning on Bi-LSTM

For Bi-LSTM, four hyper-parameters which are “look back”, “batch size”, “epochs” and “units” are considered and tuned.

Look Back (time interval of the sliding window)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Look Back	4	36.5	20.23	45	0.0042	0.0033
	8	43.7	26.86	80	0.0066	0.0055
	12	52.1	35.53	95	0.0073	0.0064

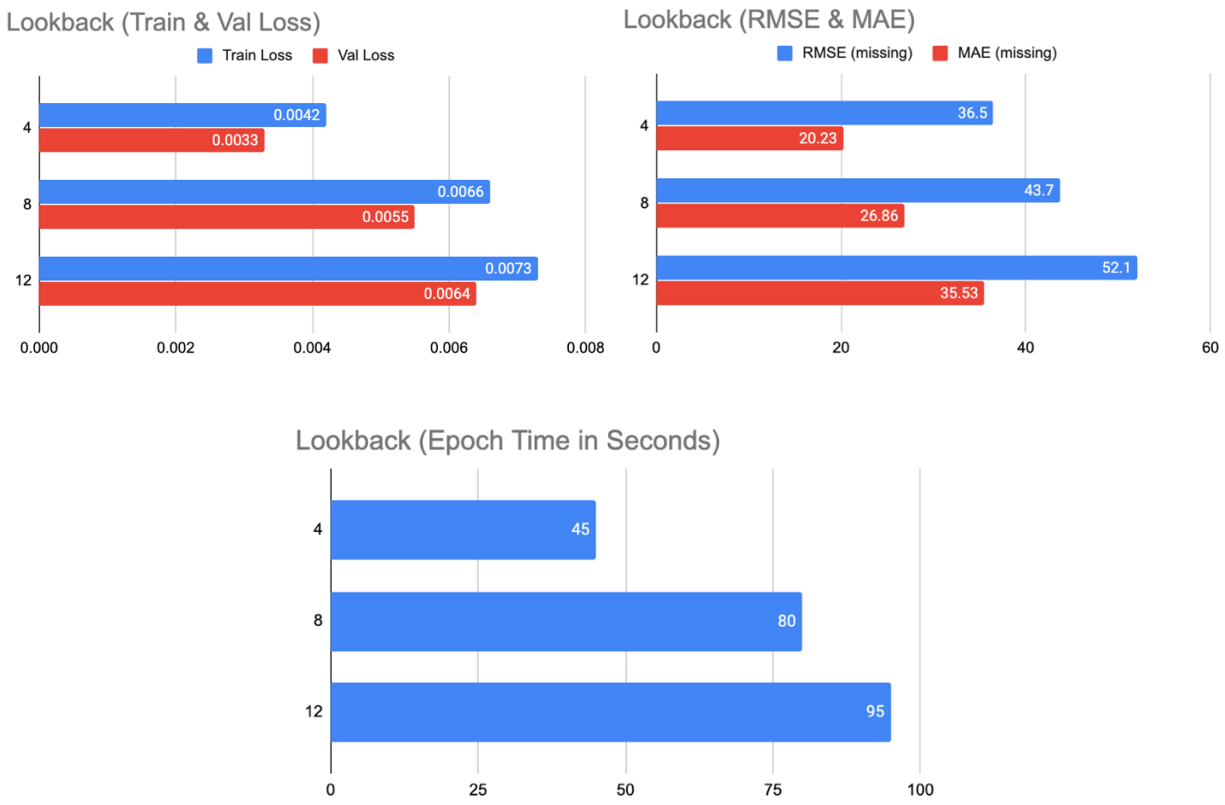

Figure 20 Tuning of look back for Bi-LSTM

As the increase of look back, both of the training loss and validation loss rise continuously, and the same for RMSE, MAE, and epoch time. The optimal lock back value seems to be 4 since it provides the smallest for all 5 metrics (Training loss: 0.0042, validation loss: 0.0033, RMSE: 36.5, MAE: 20.23, Epoch time: 45).

Batch Size (number of samples that are processed before the model is updated)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Batch Size	32	33.2	16.37	67	0.0026	0.0027
	64	35.3	19.97	30	0.0038	0.0032
	256	42.4	25.94	18	0.0051	0.0043
	512	43	26.78	16	0.0058	0.0051

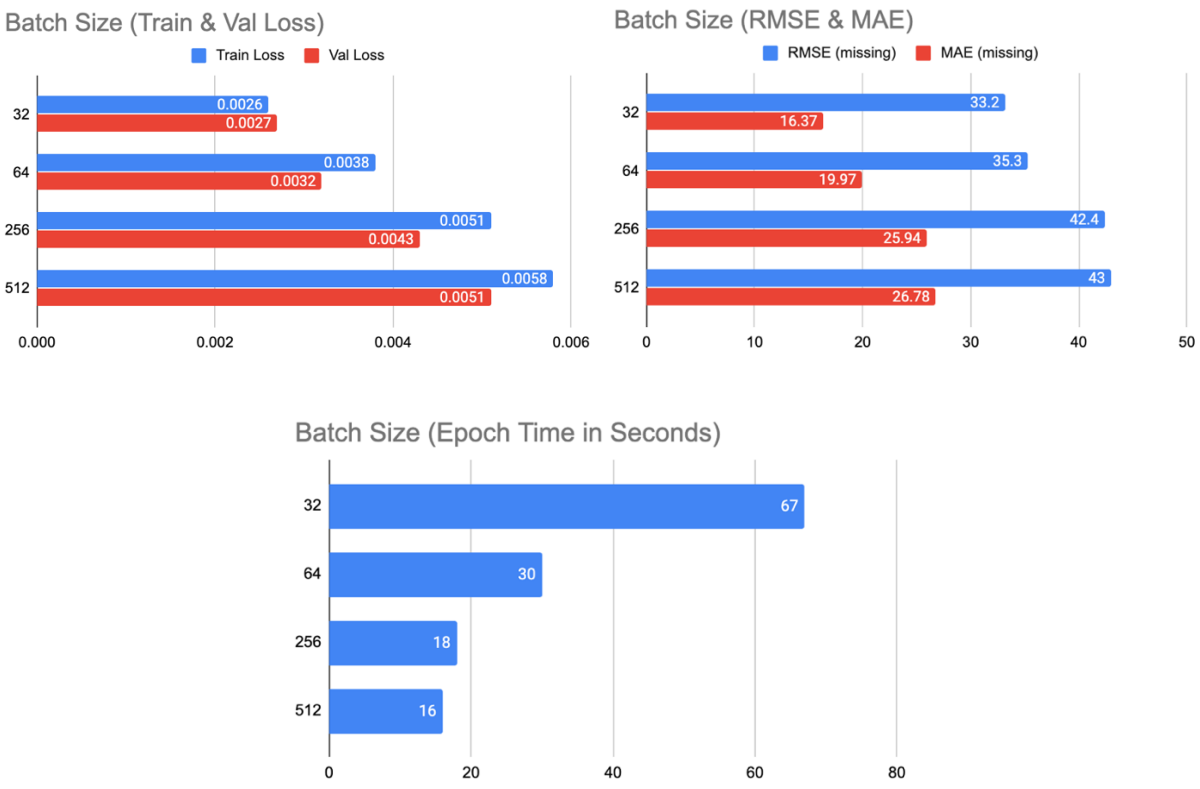

Figure 21 Tuning of batch size for Bi-LSTM

As the increase of batch size, both training and validation loss rise continuously, and the same for RMSE and MAE while epoch time drops continuously. The optimal batch size value seems to be 32 since it provides the smallest RMSE (33.2), MAE (16.37), and Validation Loss (0.0027). While the cost of computation is the largest (67) for a batch size of 32, it is worth the cost as the error and loss values are significantly better as well.

Epochs (number of the times that the dataset is used for training the model)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Epoch	5	31	16.41	74	0.0033	0.0023
	10	26.4	8.15	71	0.0024	0.0017
	20	28.3	8.36	72	0.0017	0.0019

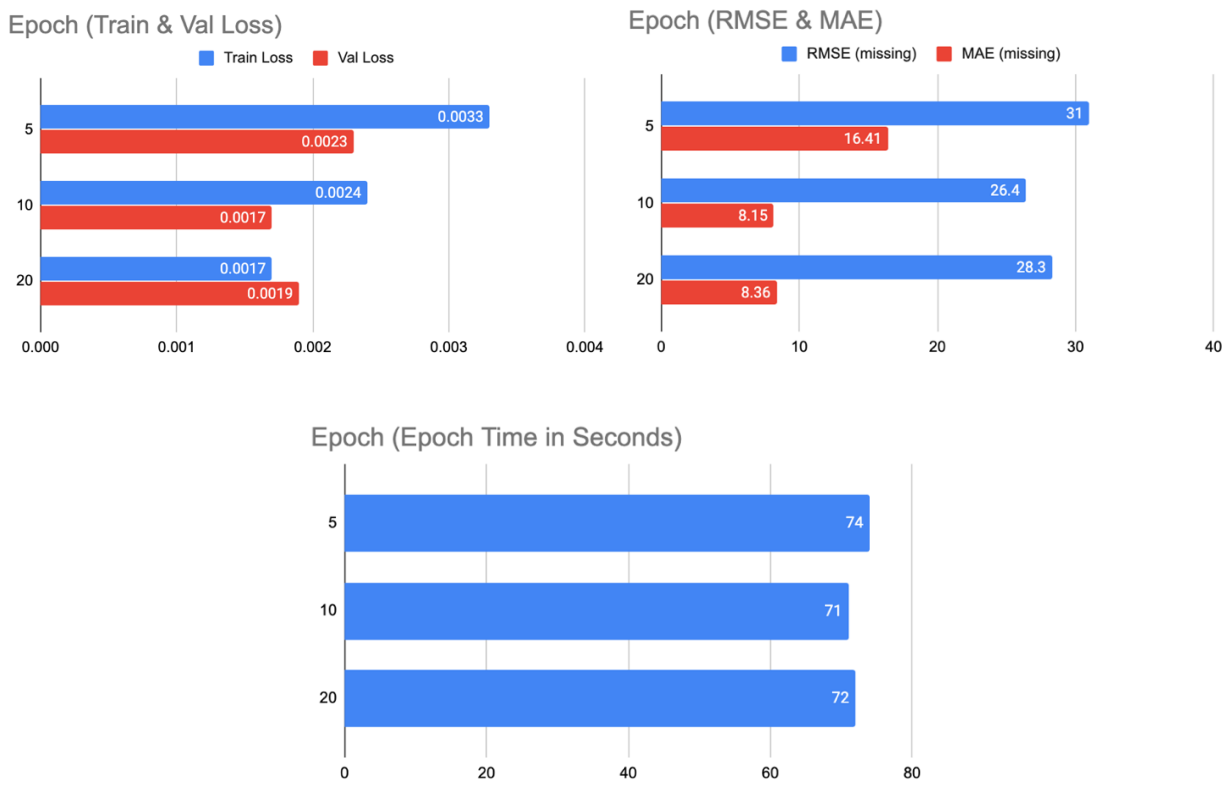

Figure 22 Tuning of epochs for Bi-LSTM

As the increase of epochs, the training loss drop continuously and validation loss first drop then rise, both RMSE and MAE first fall then rise, and the same for epoch time. The optimal epoch value seems to be 10 since it provides the smallest RMSE (26.4), MAE (8.15), and validation loss (0.0017).

Units (number of units in the LSTM layer used for feature extraction)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
LSTM Units	10	41.8	25.27	21	0.0049	0.0043
	32	35.3	19.97	30	0.0038	0.0032
	64	33.2	17.34	55	0.0034	0.0029
	128	31.2	15.42	165	0.0029	0.0025

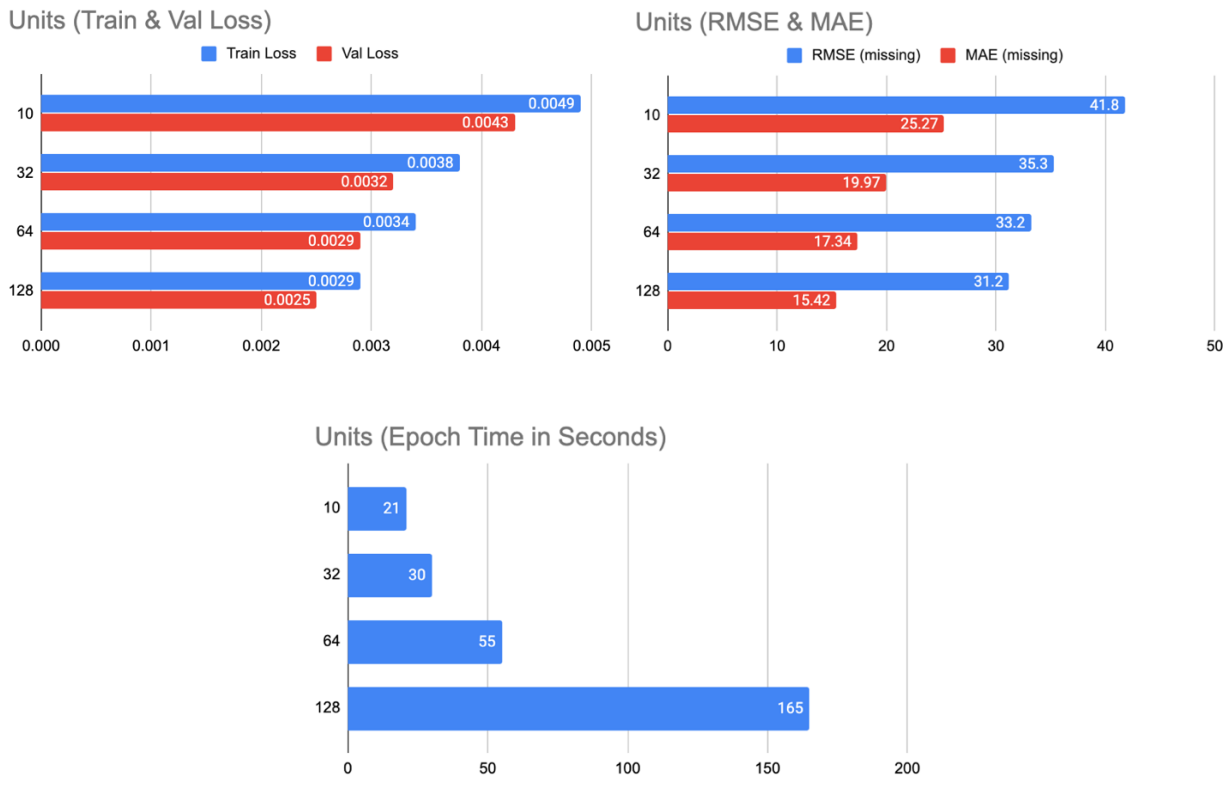

Figure 23 Tuning of units for Bi-LSTM

As the units increase, the training loss and validation loss drops, and the same for RMSE and MAE while the epoch time rises. The optimal unit value seems to be 128 since it provides the smallest RMSE (31.2), MAE (15.42), and validation loss (0.0025). While the cost of computation is the largest for a unit size of 128 (165), it is worth the cost as the error and loss values are significantly better as well.

3.4 Summary

	Model	RMSE	MAE	Epoch Time	look back	batch size	epoch
Best	W-H Avg	32.02	23.1				
	Vanilla+	29.21	9.06	0.7	8	512	200
	CNN	32.8	17.11	5	8	256	200
	BiLSTM	29	7.53	75	4	32	20

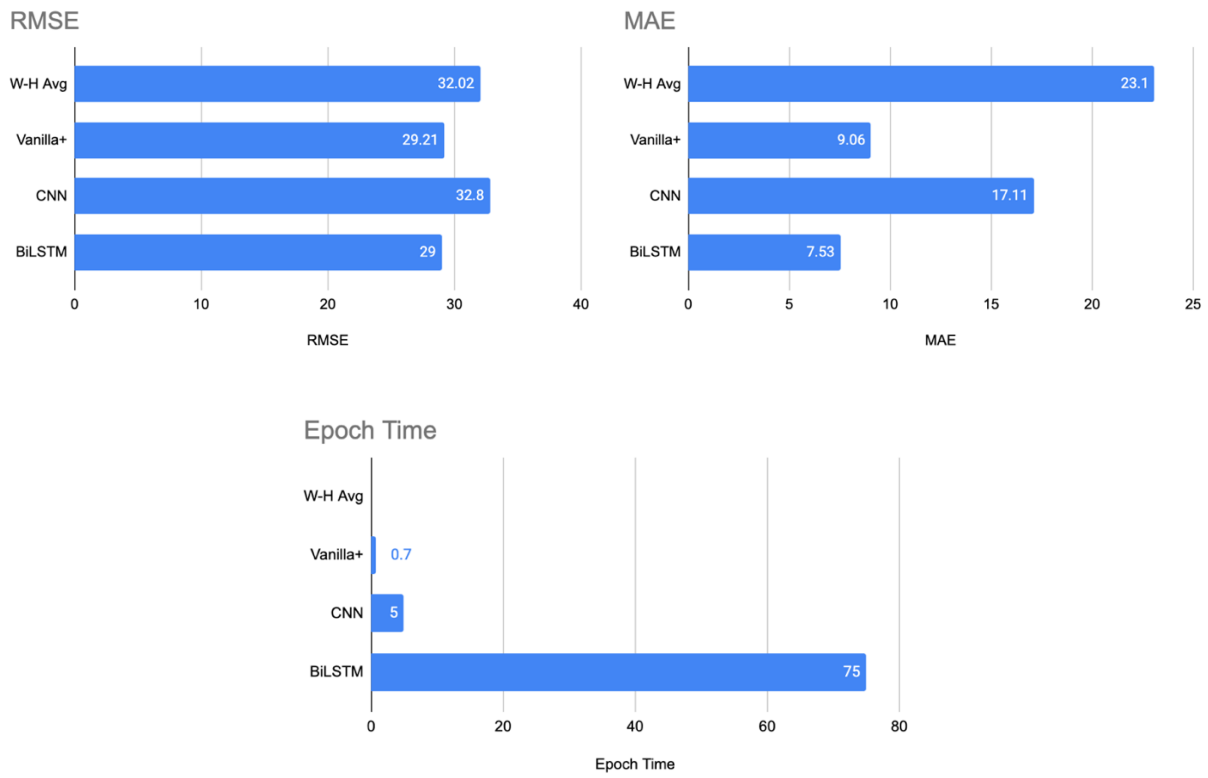

Figure 24 Summary for low missing rate dataset

The best hyper-parameter combination and result for "Vanilla", CNN and Bi-LSTM are shown above. The first model implemented was an FCN (referred to as Vanilla in this paper). They are quite old but also very simple to implement. The second model was CNN as it is good for extracting 2D features from a dataset. Finally, LSTM was used as it could look at past and future values to make an even better prediction. These three models were compared to the industry-wide accepted method of using Weekly-Hourly Average of predicting traffic flow.

Contrary to initial belief, FCN (Vanilla) model does very well in the imputation of missing data. It ties with the Bi-LSTM in RMSE and falls slightly behind in MAE. While Bi-LSMT provides the least error rate, it also takes 100x longer than FCN to compute the missing values. Even though CNN improves on the MAE of the W-H Avg, the RMSE is the worst of all

the methods compared. Providing a 2D sliding window seems to have done nothing for the CNN model here. The optimal model seems to be the tried and tested FCN.

Using a simpler model and optimizing it yielded much better results than anticipated. While Bi-LSTM has room for improvement, its complexity makes it a lot more difficult to optimize. Bi-LSTM is much more expensive computationally and yields only slightly better results than Vanilla. Optimization, then, is the key, not complexity.

CHAPTER 4: Experimental results for high missing rate

The figure below shows the trend of RMSE and MAE of different training set missing rates (from 10% to 90%) and the testing set keeps the missing rate at 15% all the time. These two values increase which means we will obtain a larger error as the missing rate increases.

Missing rate	RMSE	MAE
10%	38.9	15.3
20%	32.4	14.5
30%	33.5	15.6
40%	32.7	16.1
50%	33	17.4
60%	33.3	19
70%	37.3	22.7
80%	35.5	21.1
90%	38.1	24.3

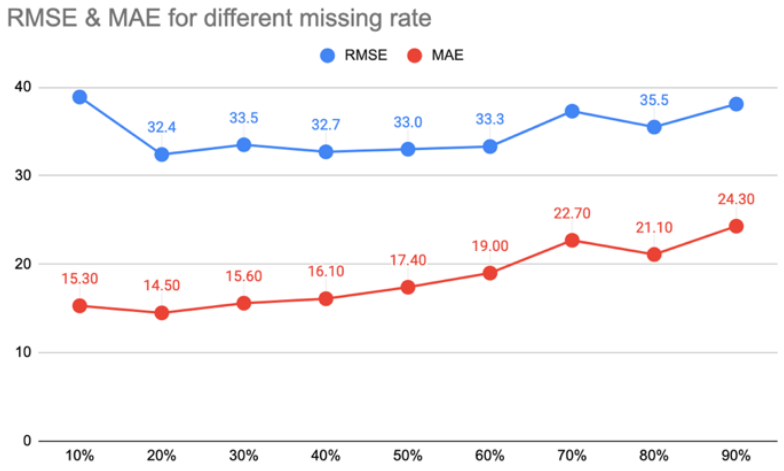

Figure 25 RMSE and MAE of different missing rates

In this section, the missing rate of 80% is used for generating training data and 15% for testing (validation data) which are shown in the figure as follows. And this high missing rate data is used to verify the robustness of the Autoencoder model under the most extreme situations.

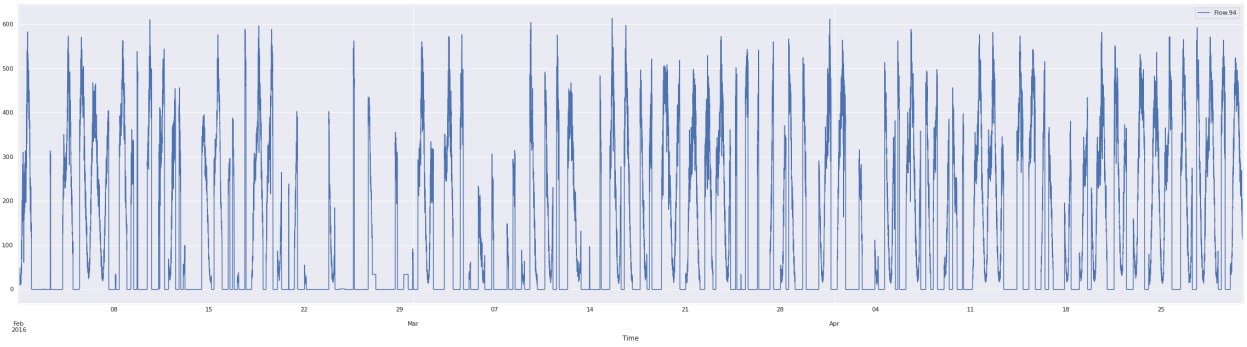

Figure 26 Training dataset with about 80% missing rate (high)

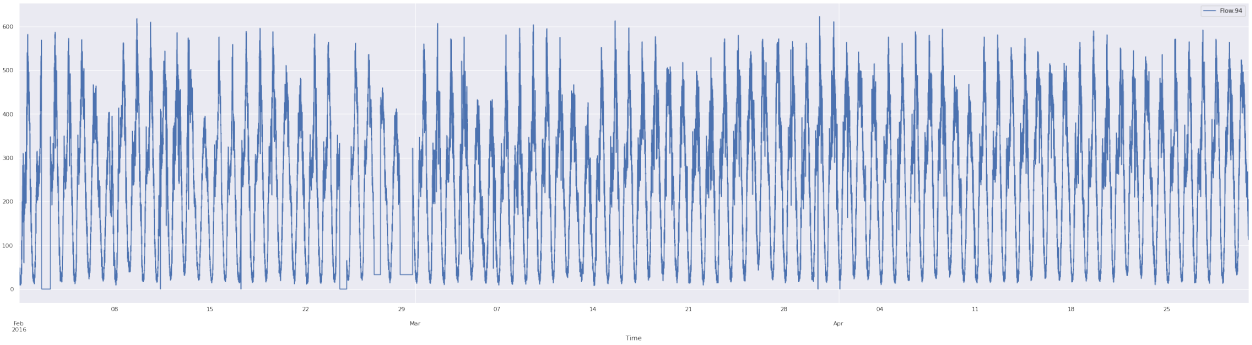

Figure 27 Complete training dataset

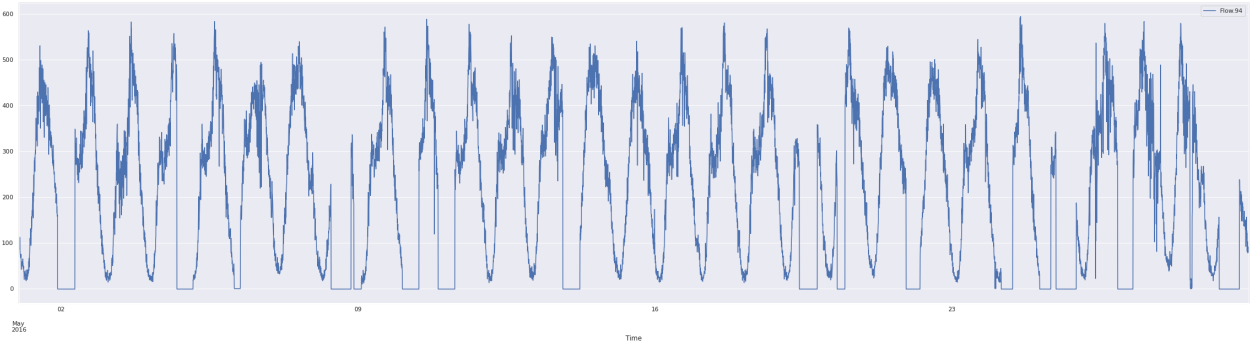

Figure 28 Testing dataset with 15% missing rate

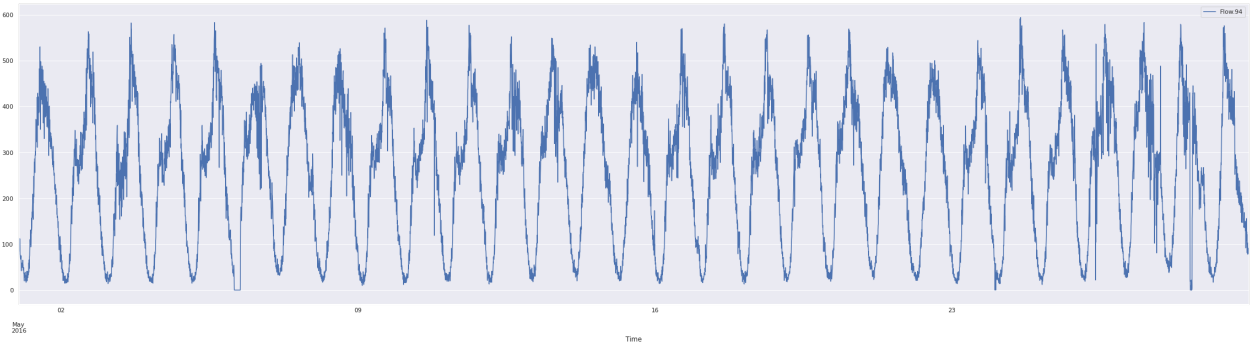

Figure 29 Complete testing dataset

In this section, three models: “Vanilla”, CNN, and Bi-LSTM are considered and their hyper-parameter tuning results are shown as follows.

4.1 Effects of hyperparameter tuning on “Vanilla” (FCN)

For “Vanilla”, four hyper-parameters which are “look back”, “batch size”, “epochs”, and “units” are considered and tuned.

Look Back (time interval of the sliding window)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Look Back	4	30.24	14.28	0.2	0.0031	0.0024
	8	31.2	15.05	0.2	0.0029	0.0033
	16	33.06	17.23	0.3	0.0028	0.0028

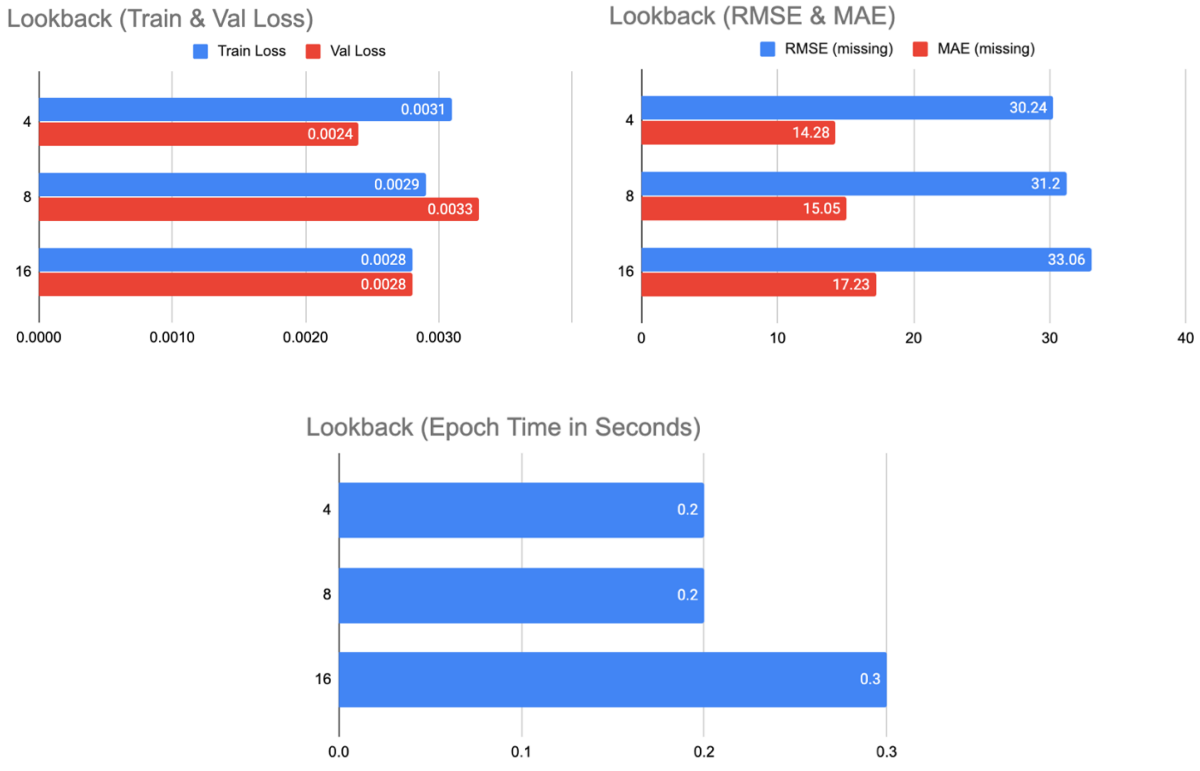

Figure 30 Tuning of look back for “Vanilla”

As the increase of look back, training loss falls continuously while validation loss first rises then falls, both RMSE and MAE increase continuously, and the epoch time is almost the same. The optimal lookback value seems to be 4 since it provides the smallest RMSE (30.24) and MAE (14.28) values. Validation Loss for a look back of 4 (0.0024) is also the smallest.

Batch Size (number of samples that are processed before the model is updated)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Batch Size	64	26.97	10.27	0.4	0.0021	0.0019
	128	28.27	11.32	0.4	0.0022	0.002
	256	27.65	11.14	0.3	0.0025	0.002
	512	29.7	13.25	0.3	0.0028	0.0022
	1024	31.55	15.15	0.3	0.0031	0.0025

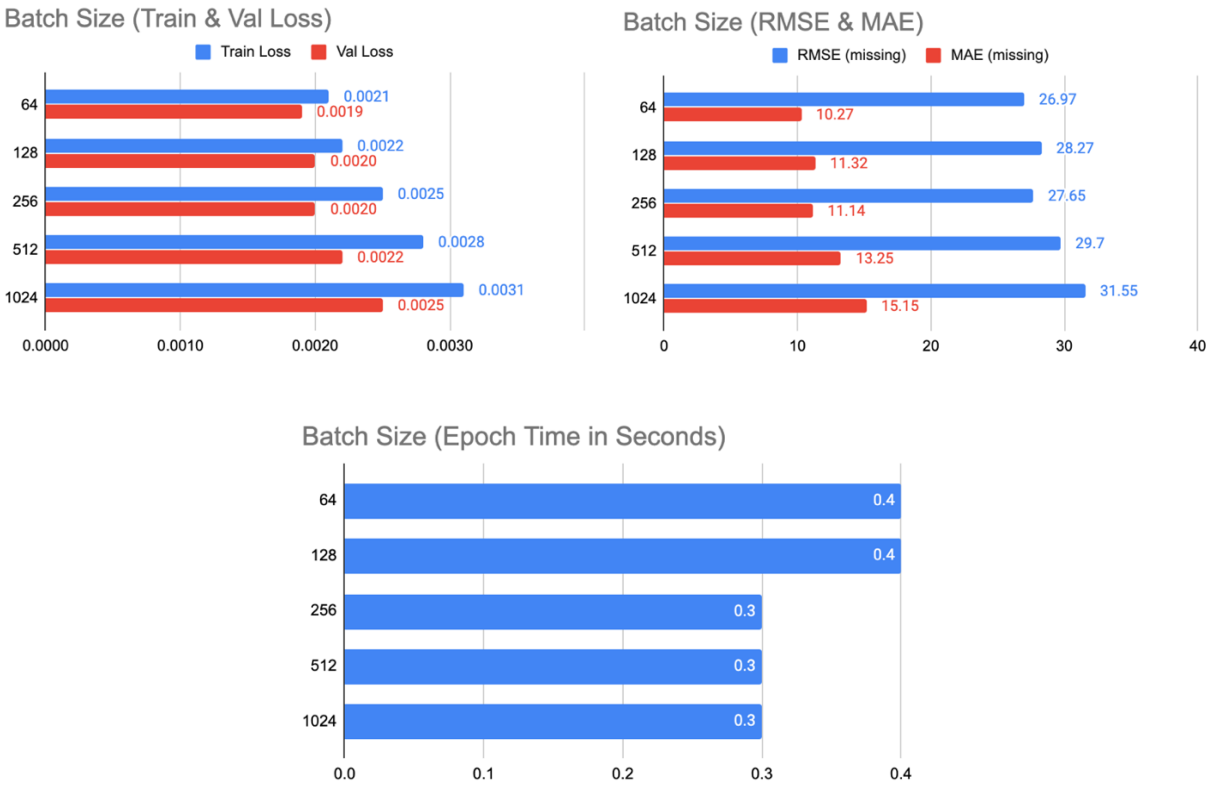

Figure 31 Tuning of batch size for "Vanilla"

As the batch size increases, both training loss and validation loss increase, and almost the same for RMSE and MAE. All the epoch times are almost the same. The optimal batch size

value seems to be 64 since it provides the smallest RMSE (26.97) and MAE (10.27) values. And the result of training and validation loss (0.0021, 0.0019) also verifies that batch size 64 is the best.

Epochs (number of the times that the dataset is used for training the model)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Epoch	25	31.42	15.85	0.5	0.0025	0.0025
	50	26.97	10.27	0.4	0.0021	0.0019
	100	26.87	9.76	0.5	0.0019	0.0019
	200	26.77	9.55	0.5	0.0017	0.0018
	300	26.97	9.62	0.5	0.0015	0.0018

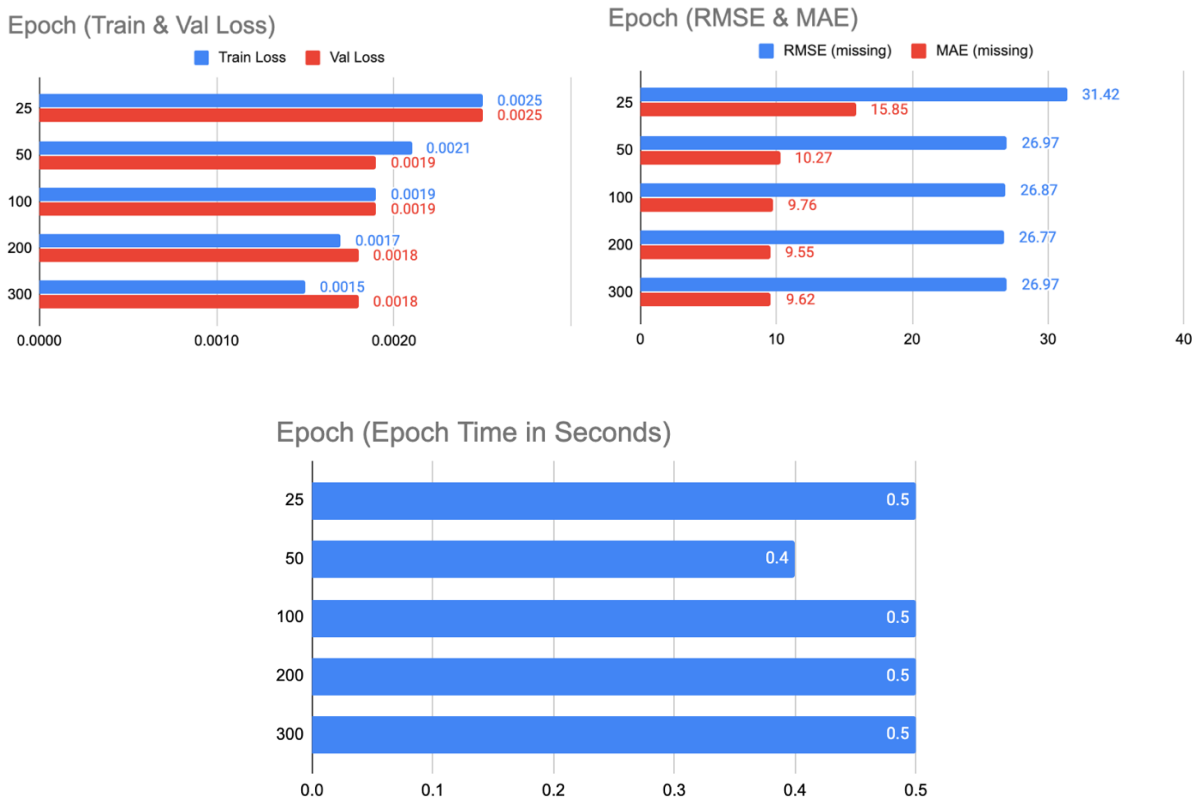

Figure 32 Tuning of epochs for "Vanilla"

As the epochs increase, both training loss and validation loss fall, and almost the same is observed for RMSE and MAE. The epoch time is almost the same. The optimal epoch value

seems to be 200 rounds since it provides the smallest RMSE (26.77) and MAE (9.55). Even though the epoch of 300 rounds has similar RMSE and MAE, the 200 will save much time.

Units (number of units in the hidden layer used for feature extraction)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Units	64	31.72	16.96	0.3	0.0037	0.0026
	128	30.14	15.16	0.3	0.0031	0.0024
	256	29.14	13.53	0.3	0.0027	0.0022
	512	29.11	13.26	0.3	0.0023	0.0021
	1024	26.97	10.27	0.4	0.0021	0.0019
	2048	27.92	10.8	0.9	0.002	0.002

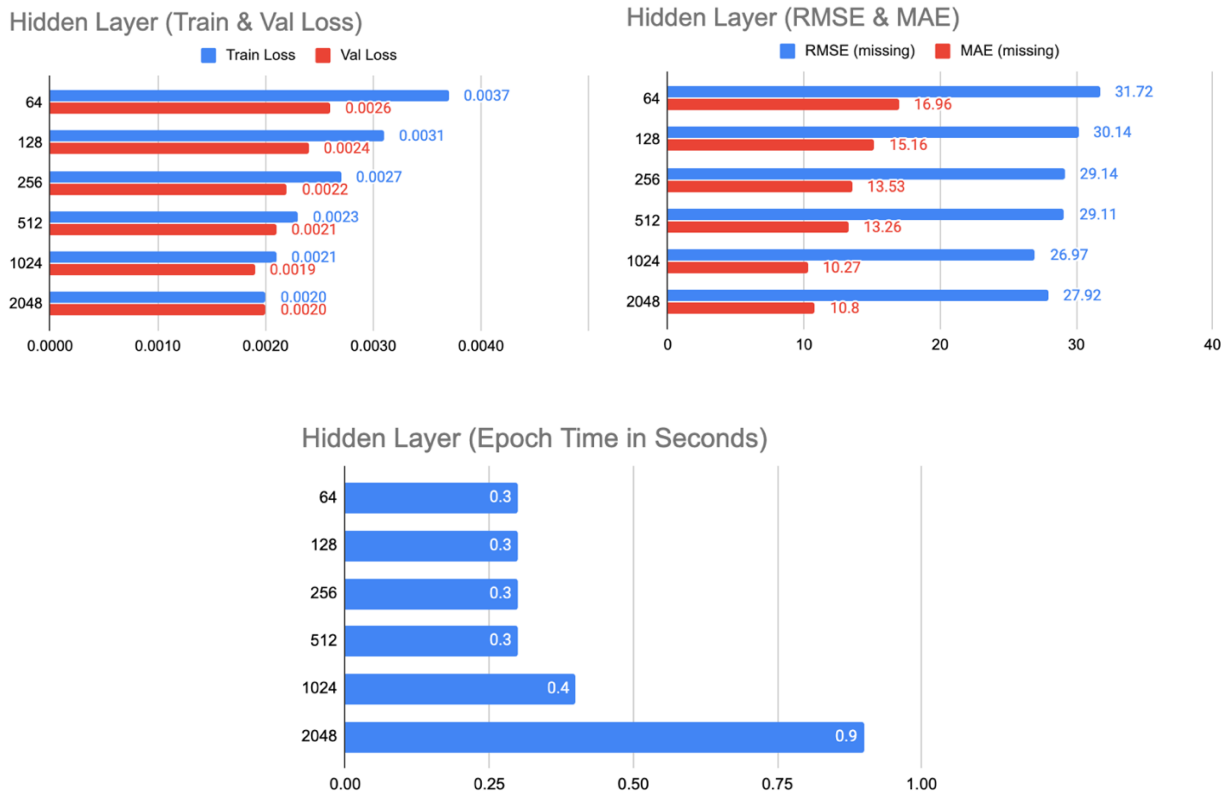

Figure 33 Tuning of units for "Vanilla"

As the increase of units, both training loss and validation loss fall continuously, and nearly the same for RMSE and MAE. But the epoch time of unit 2048 is almost three times of unit 64 to 512. The optimal unit value seems to be 1024 since it provides the smallest RMSE

(26.97) and MAE (10.27) with the smallest training loss (0.0021) and validation loss (0.0019). And the epoch time (0.4) is also acceptable.

4.2 Effects of hyperparameter tuning on CNN

For CNN, four hyper-parameters which are “look back”, “batch size”, “epochs”, and “structure” are considered and tuned.

Look Back (time interval of the sliding window)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Look Back	4	36.7	21.15	0.8	0.5418	0.5467
	8	39.95	24.65	1.2	0.5441	0.5483
	16	39.95	24.3	2.2	0.5437	0.548

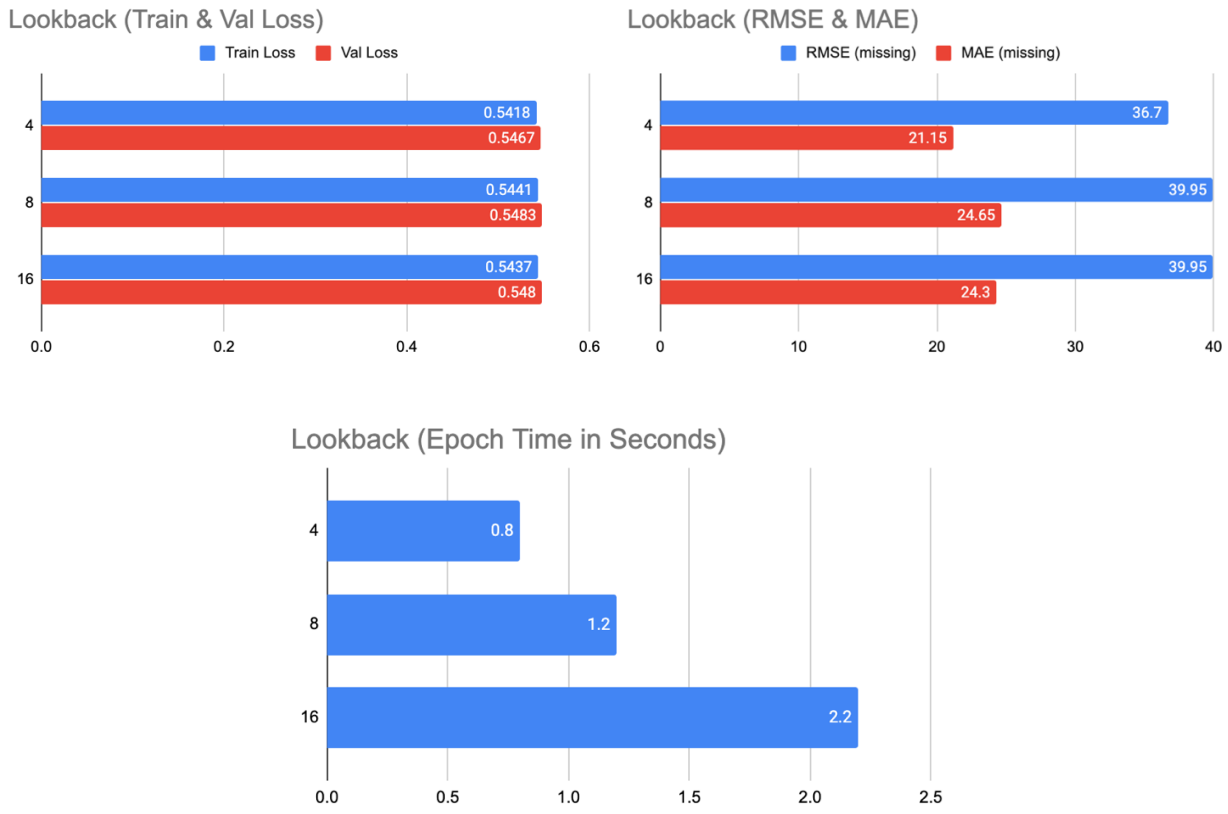

Figure 34 Tuning of look back for CNN

As the look back increases, both training loss and validation loss are nearly unchanged, RMSE and MAE and epoch time all increase. The optimal look back value seems to be 4 due to the smallest RMSE (36.7), MAE (21.15), and epoch time (0.8).

Batch Size (number of samples that are processed before the model is updated)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Batch Size	64	36.41	21.12	1.2	0.5437	0.5468
	128	36.97	22.28	1	0.5426	0.5474
	256	36.7	21.15	0.8	0.5418	0.5467
	512	40.08	25.36	0.7	0.5432	0.5482

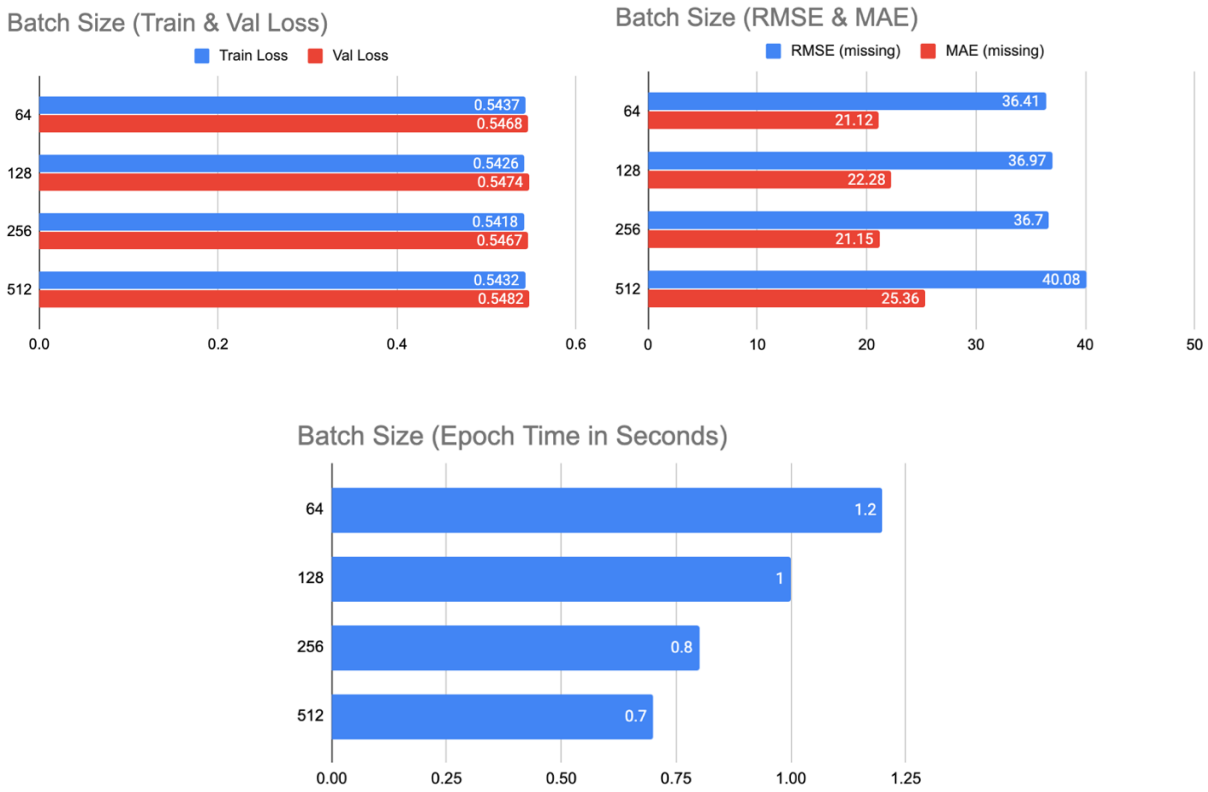

Figure 35 Tuning of batch size for CNN

As the batch size increases, both training loss and validation loss are nearly unchanged, and both RMSE and MAE fluctuate while the epoch time falls. The optimal batch size value seems to be 256 due to the smallest RMSE (36.7) and MAE (21.15), and the epoch time of 256 (0.8) is also acceptable.

Epochs (number of the times that the dataset is used for training the model)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Epoch	25	37.36	21.8	0.9	0.5426	0.5471
	50	36.7	21.15	0.8	0.5418	0.5467
	100	36.46	21.39	0.8	0.5419	0.5468
	200	37.22	21.36	0.8	0.5424	0.5469

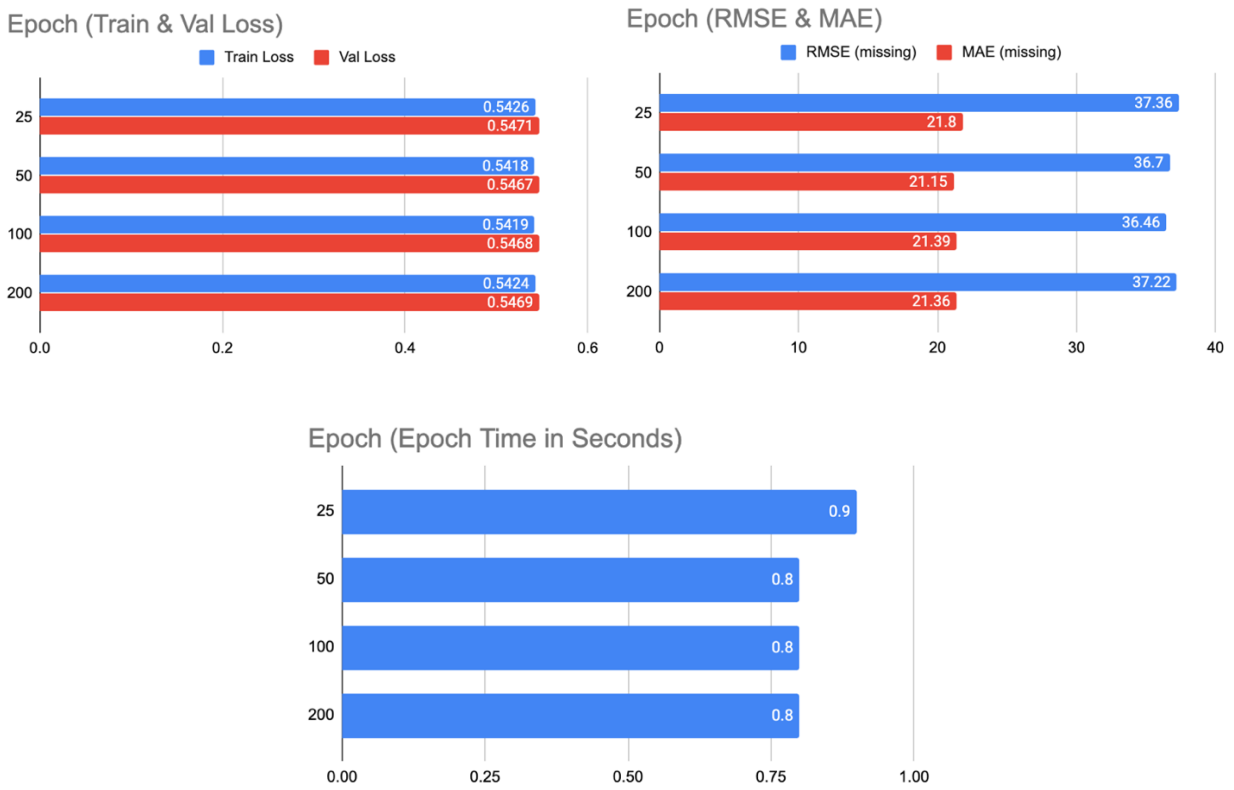

Figure 36 Tuning of epochs for CNN

As the epochs increase, the training loss, validation loss, RMSE, and MAE remain unchanged. The optimal epoch value seems to be 50 since it provides the smallest MAE (21.15) value, and the epoch time for 50 (0.8) is also acceptable.

Structure

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Structure	2 1 1 2	51.17	34.44	0.7	0.5499	0.5544
	4 2 2 4	36.7	21.15	0.8	0.5418	0.5467
	8 4 4 8	33.9	19.07	0.9	0.5409	0.5457
	16 8 8 16	37	20.25	1.1	0.5392	0.5465

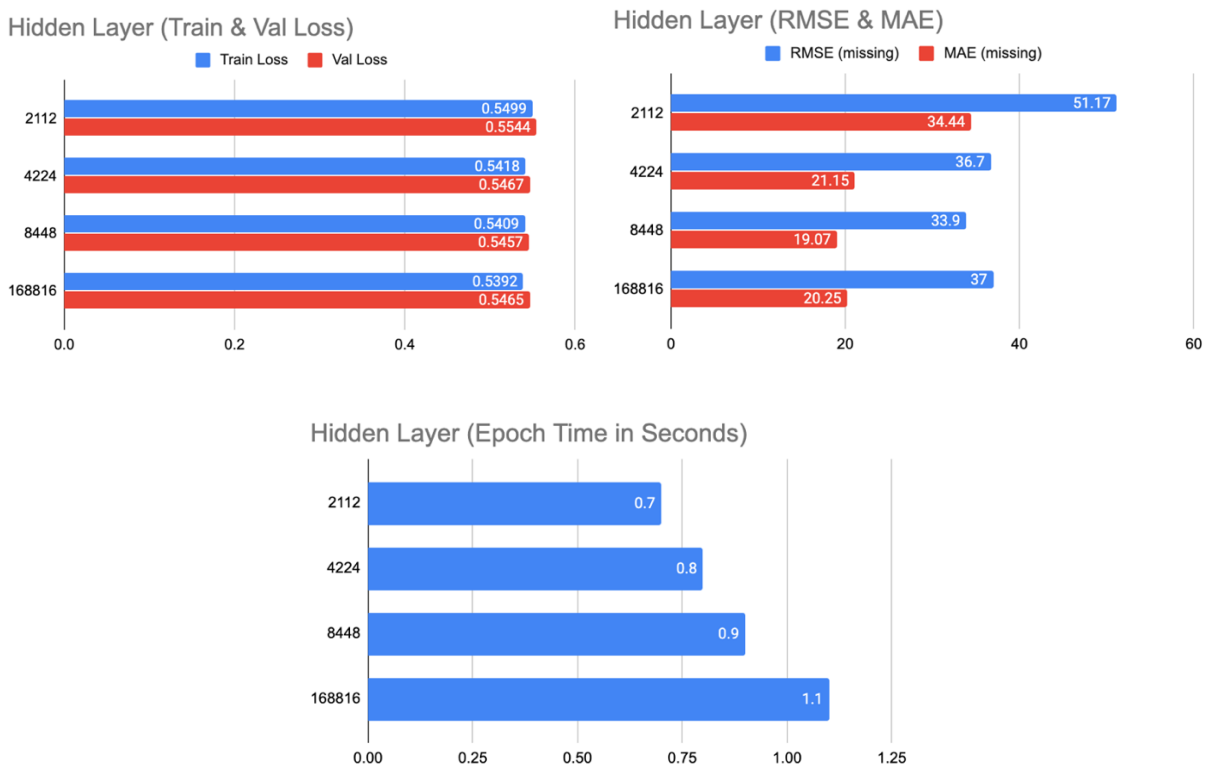

Figure 37 Tuning of structure for CNN

With an increase of structure complexity, both training loss, and validation loss remain unchanged, RMSE and MAE first fall then rise, and the epoch time rises. From the figure shown above, the 8-4-4-8 structure is the best since it has the lowest RMSE (33.9) and MAE (19.07), and the epoch time (0.9) is also acceptable.

4.3 Effects of hyperparameter tuning on Bi-LSTM

For Bi-LSTM, four hyper-parameters which are “look back”, “batch size”, “epochs” and “units” are considered and tuned.

Look Back (time interval of the sliding window)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Look Back	4	27.58	10.82	39	0.0044	0.0019
	8	26.9	9.44	80	0.0042	0.0018
	16	31.52	16.49	145	0.0061	0.0025

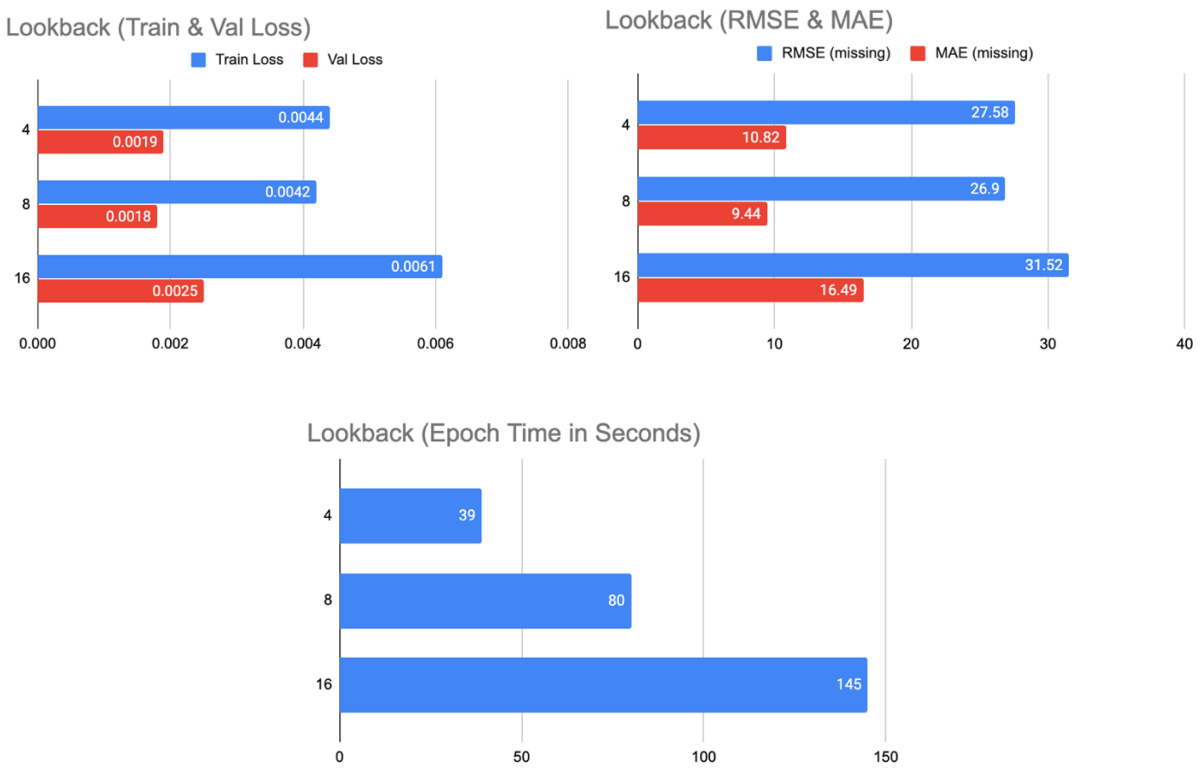

Figure 38 Tuning of look back for Bi-LSTM

As the look back increases, training loss and validation loss first fall then rise, and the same trend is seen for RMSE and MAE while the epoch time increases. The optimal lock back value seems to be 8 since the training loss (0.0042) and validation loss (0.0018) are both the lowest, and the same for RMSE (26.9) and MAE (9.44). The epoch time of 8 is also acceptable, and it is worth the cost as the error and loss values are significantly better.

Batch Size (number of samples that are processed before the model is updated)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Batch Size	32	26.9	9.44	80	0.0042	0.0018
	64	27.89	11.01	50	0.0047	0.0019
	128	30.18	14.21	39	0.0052	0.0023

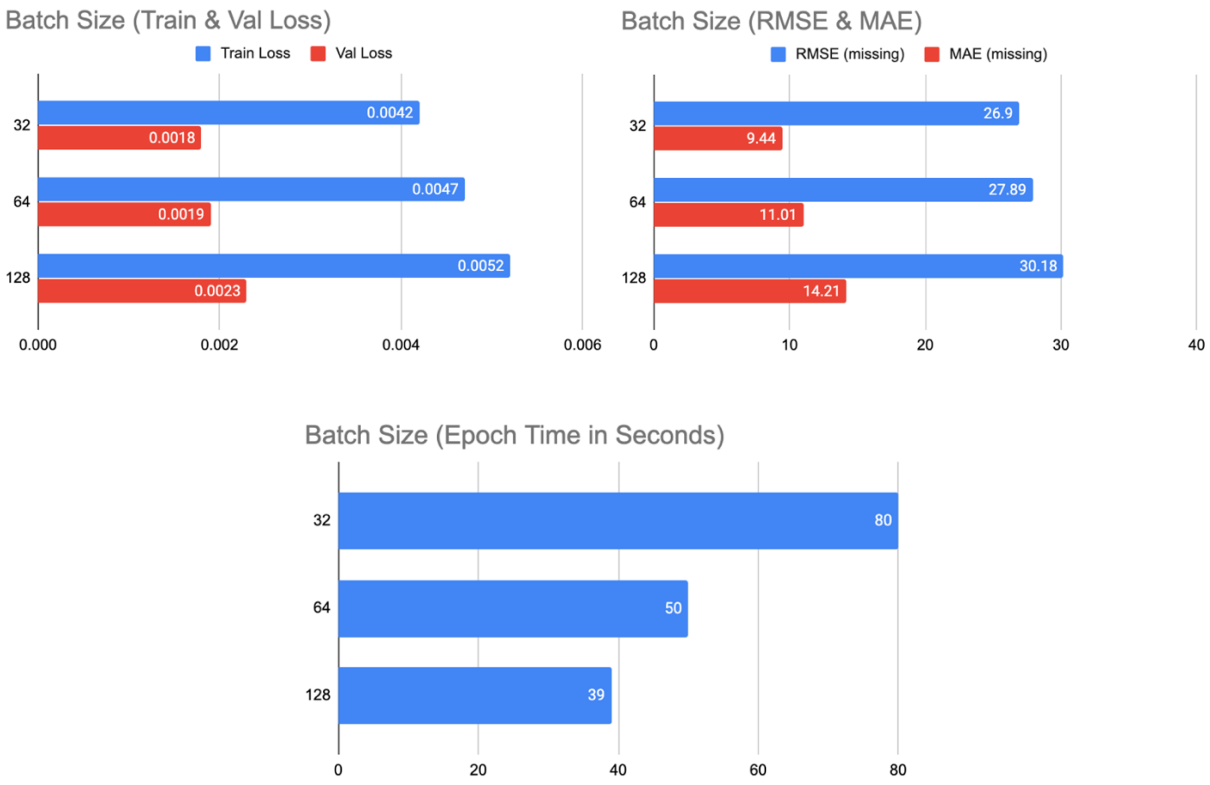

Figure 39 Tuning of batch size for Bi-LSTM

As the batch size increases, training loss, validation loss, RMSE, and MAE all rise, while the epoch time drops rapidly. The optimal batch size value seems to be 32 due to the best RMSE (26.9) and MAE (9.44). Even though the epoch time for 32 is the longest (80), it is worth the cost as the error and loss values are significantly better as well.

Epochs (number of the times that the dataset is used for training the model)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
Epoch	25	26.9	9.44	80	0.0042	0.0018
	50	25.85	8.41	76	0.0035	0.0016
	100	27.57	11.56	74	0.0047	0.0022

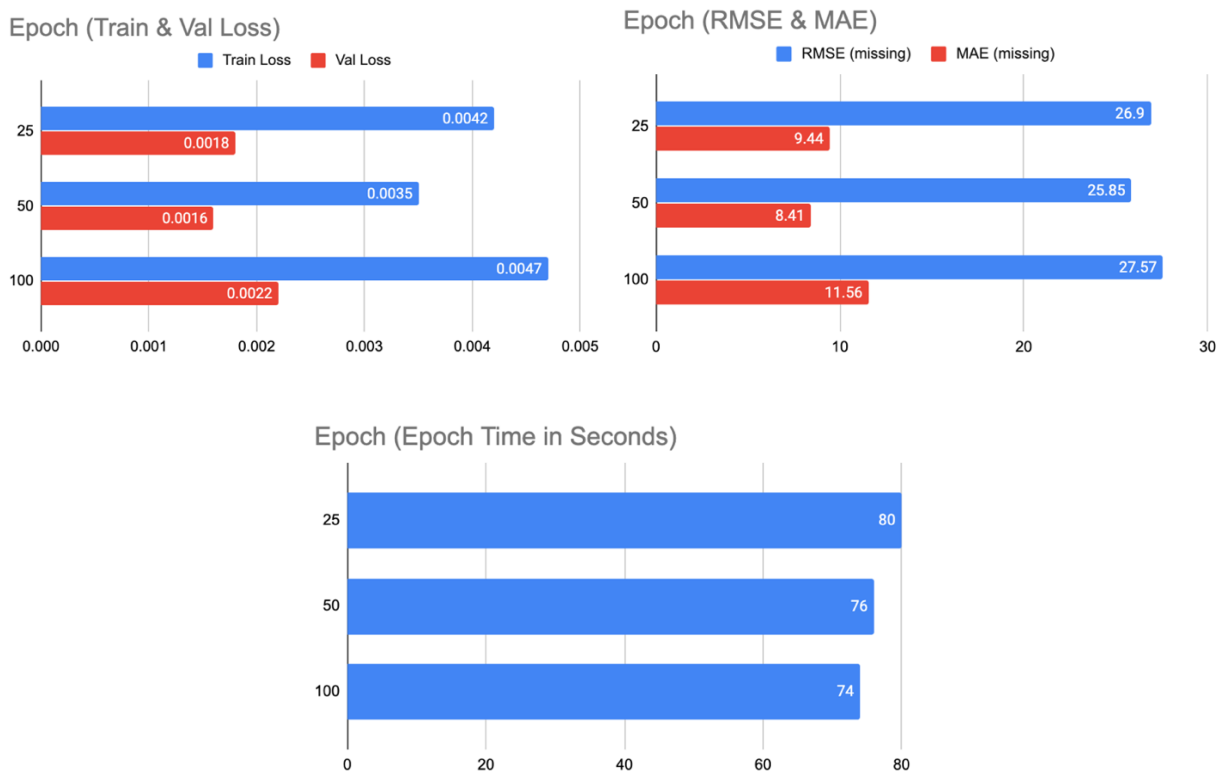

Figure 40 Tuning of epochs for Bi-LSTM

As the epochs increase the training loss and validation loss first drop then rise, and the same trend is true for RMSE and MAE while the epoch time drops. The optimal epoch value seems to be 50 because of the lowest RMSE (25.85) and MAE (8.41). And the epoch time of 50 is also acceptable and it is worth the cost as the error and loss values are significantly better as well.

Units (number of units in the LSTM layer used for feature extraction)

Parameter		RMSE (missing)	MAE (missing)	Epoch Time	Train Loss	Val Loss
LSTM Units	8	35.3	22.34	55	0.0063	0.003
	16	40.76	26.53	60	0.0078	0.0043
	32	26.9	9.44	80	0.0042	0.0018
	64	25.94	8.26	122	0.0036	0.0017
	128	26.05	8.59	350	0.0036	0.0017

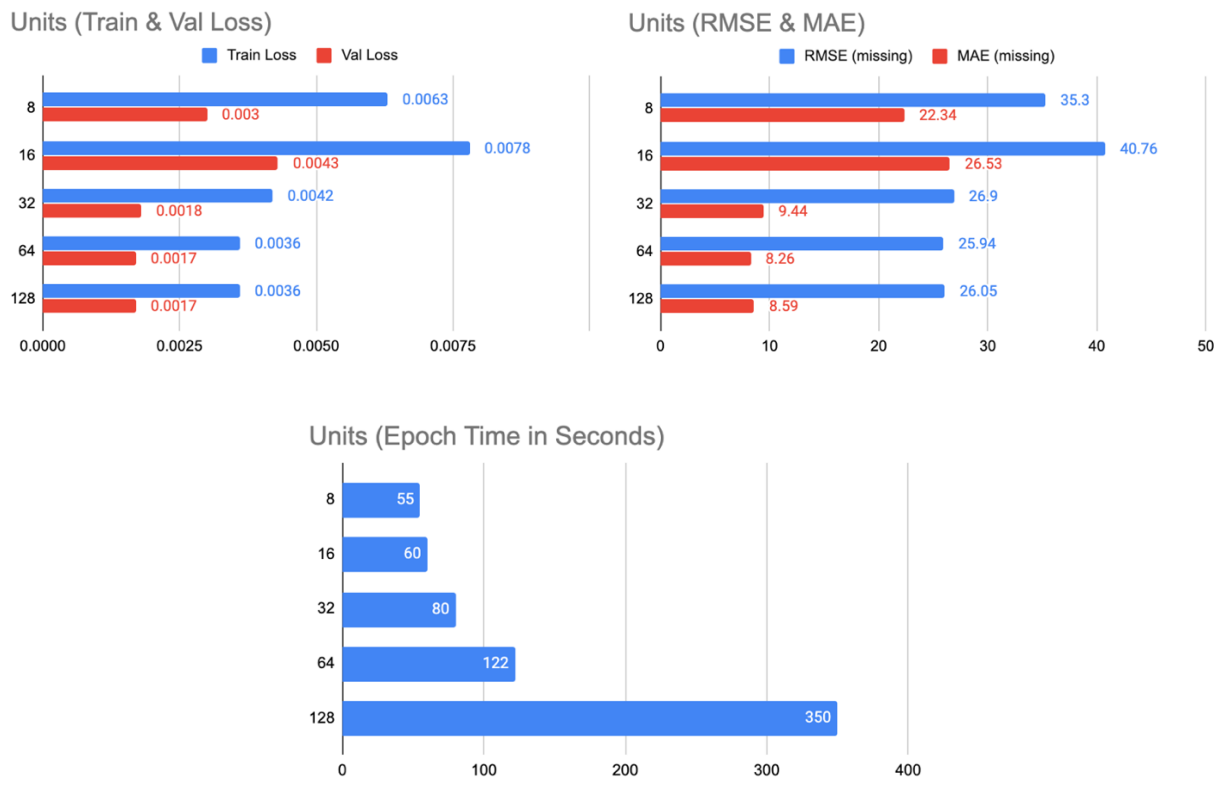

Figure 41 Tuning of units for Bi-LSTM

As the epochs increase the training loss and validation loss first rises then drops, and the same trend is seen for RMSE and MAE. The epoch time rises continuously. The optimal unit value seems to be 64 because of the lowest RMSE (25.94) and MAE (8.26). And the epoch time of 64 is also acceptable and it is worth the cost as the error and loss values are significantly better as well.

4.4 Summary

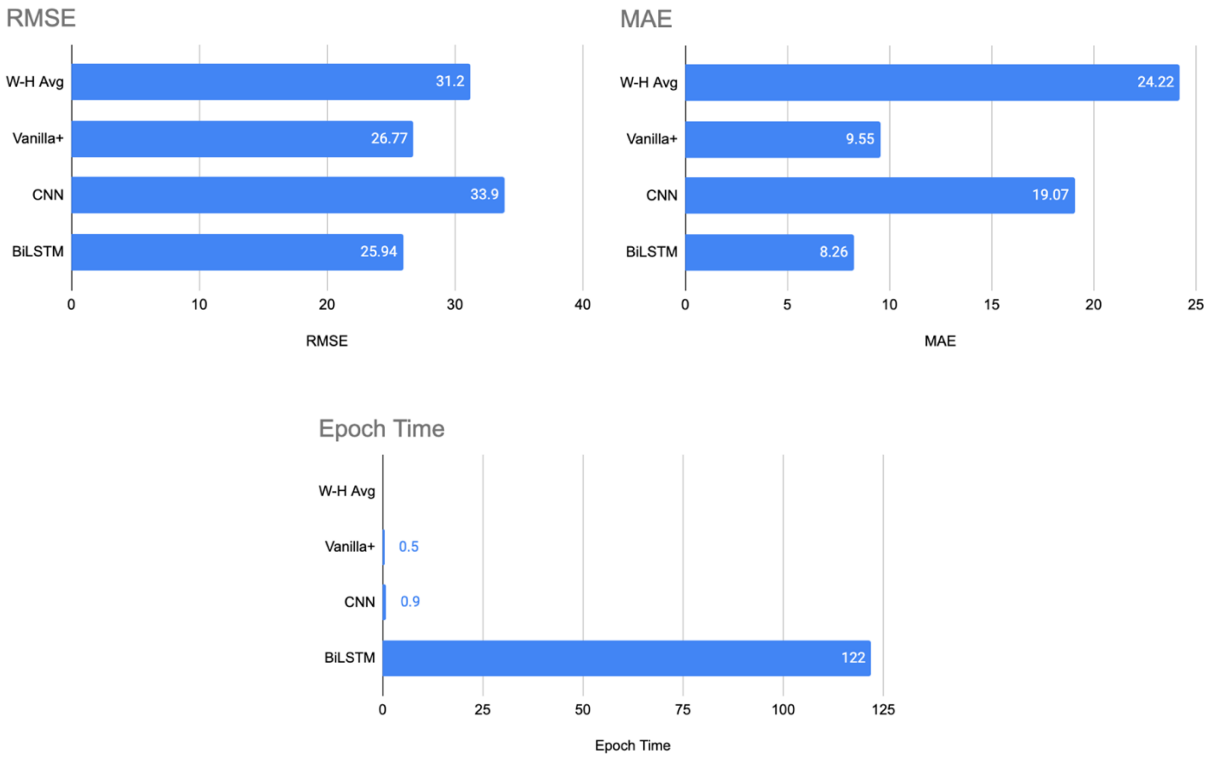

Figure 42 Summary for high missing rate dataset

The best hyper-parameter combination and result for "Vanilla", CNN, and Bi-LSTM to process the high missing rate data are shown above. These three models were compared to the industry-widely accepted method of using Weekly-Hourly Average of predicting traffic flow.

Even for the high missing rate dataset, the simplest model, FCN (Vanilla), also does very well in the imputation of missing data. It ties with the Bi-LSTM in RMSE and falls slightly behind in MAE. While Bi-LSMT provides the least error rate, it also takes much longer than FCN to compute the missing values. And in the high missing rate case, the time is much longer than the low missing rate case. Even though CNN improves on the MAE of the W-H Avg, the RMSE is the worst of all the methods compared. As same as the low missing rate case,

providing a 2D sliding window seems to have done nothing for the CNN model here. The optimal model seems to be the tried and tested FCN.

Using a simpler model and optimizing it yielded much better results than anticipated. While Bi-LSTM has room for improvement, its complexity makes it a lot more difficult to optimize. Bi-LSTM is much more expensive computationally and yields only slightly better results than “Vanilla”.

CHAPTER 5: Error patterns for different stations and hours

In Chapters 3 and 4, the whole dataset is considered while calculating the RMSE and MAE. In this chapter the data from different stations or different hours are separated and calculated independently for RMSE and MAE.

5.1 Error pattern for different stations

First, the data are separated according to the sensor stations and calculated respectively. The results are shown as follows.

RMSE			
Stations	Vanilla+	CNN	BiLSTM
94	23.77	36.24	27.18
95	21.78	30.49	16.22
97	13.51	21.26	11.62
100	13.25	21.37	10.35
101	17.92	24.56	10.74
104	62.44	66.07	64.23
106	18.19	30.28	14.19
107	29.14	48.22	22.76

MAE			
Stations	Vanilla+	CNN	BiLSTM
94	12.21	21.34	13.87
95	12.10	19.13	9.82
97	8.91	15.20	7.03
100	9.52	15.70	7.09
101	12.63	18.08	7.06
104	18.43	27.85	17.64
106	11.60	22.14	8.56
107	16.52	37.45	14.12

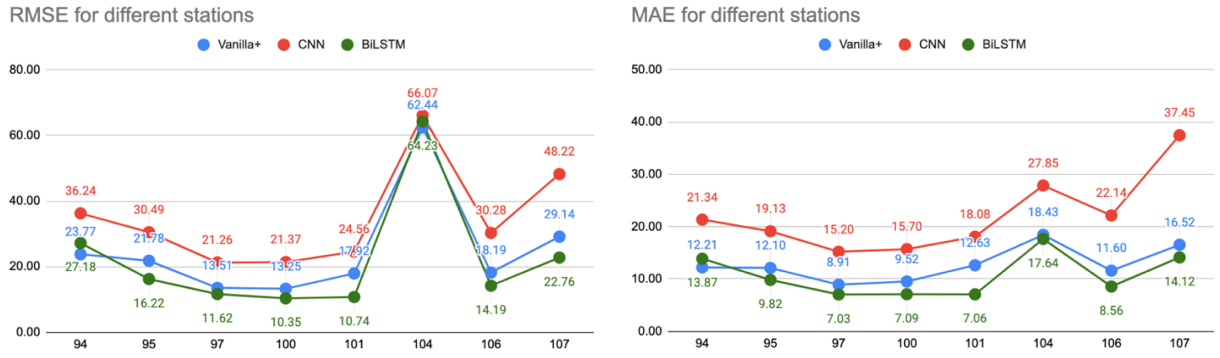

Figure 43 RMSE and MAE for different stations

From the figure above, all three models show similar trends. The Bi-LSTM model has the best performance compared with the "Vanilla" and CNN, and it can verify the conclusion that Bi-LSTM has the lowest RMSE and MAE in the last section. Two stations: 106 and 107 have the largest error value especially for 104 with the highest RMSE of about 63 no matter which model is used. Several reasons may cause this problem.

a) Facility error

The measurement results may be inaccurate due to loop detector malfunctions of various kinds. So the value station 104 may be erratic and chaotic, and it's hard for a machine learning model to identify the traffic flow patterns.

b) Chaotic operation

Due to less than ideal geometric design or extremely large traffic volume, traffic congestion because of accidents or large volumes may frequently occur near station 104, and it may be hard to find a regular pattern especially in the case of accidents caused by problematic road design.

Because of the lack of a sensor location distribution map, it is hard to track the exact problems. A future study could focus on the details of it.

5.2 Error patterns for different hours

In this part, the data of each hour is separated, and RMSE and MAE are calculated respectively as shown below.

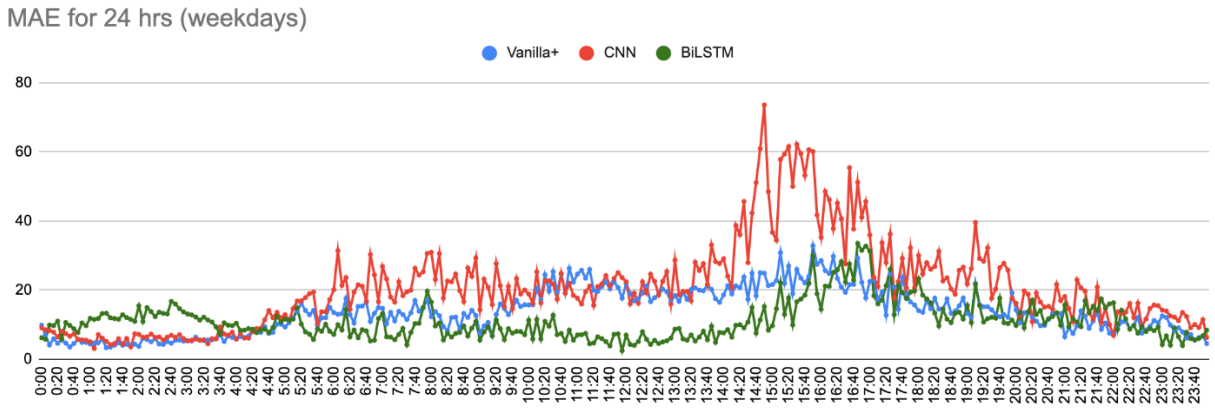

Figure 44 MAE for 24 hours (weekdays)

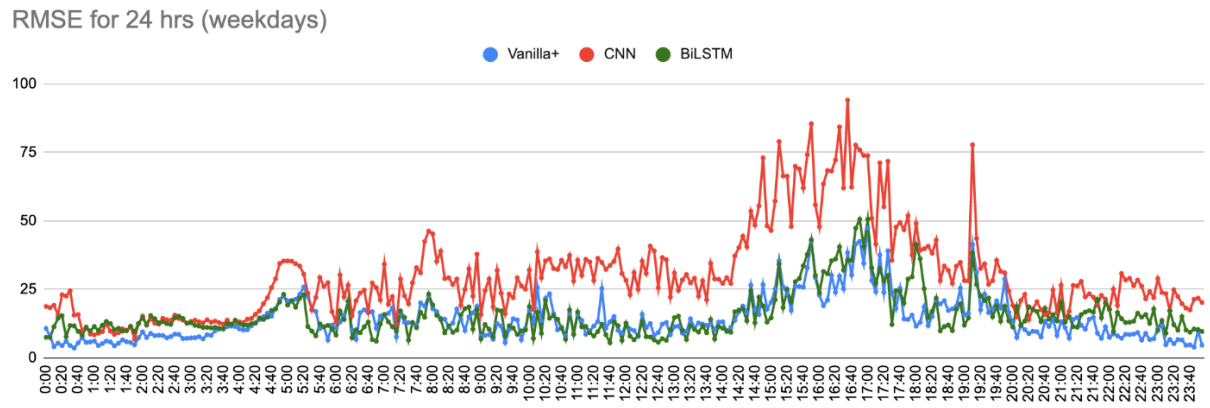

Figure 45 RMSE for 24 hours (weekdays)

As shown in Figures 44 and 45, we can see the error patterns for 24 hours on weekdays. All three models show a similar error trend. And it can verify the conclusion in the previous section that the "Vanilla" and Bi-LSTM are better than CNN. Besides, two error peaks are concentrated on the morning peak hour (around 8:00) and afternoon peak hour (17:00) which are consistent with the daily flow peak hours.

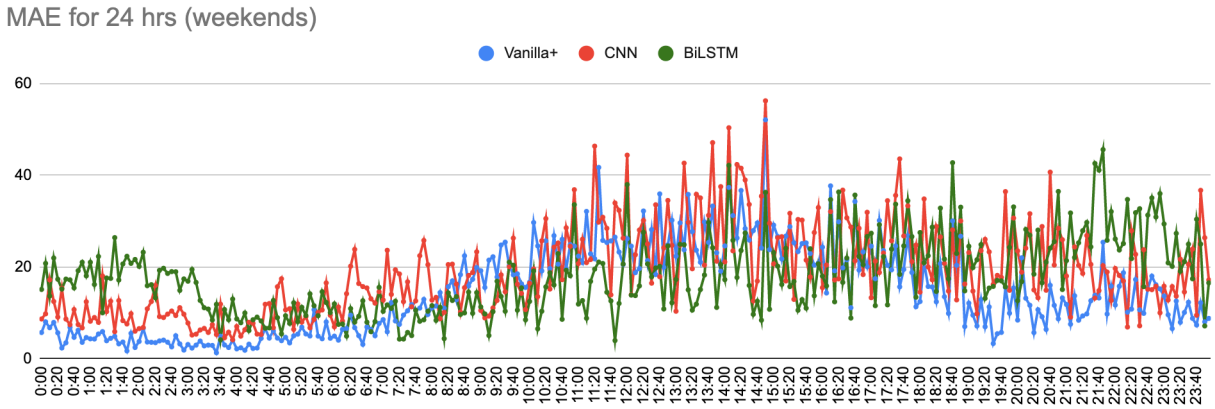

Figure 46 MAE for 24 hours (weekends)

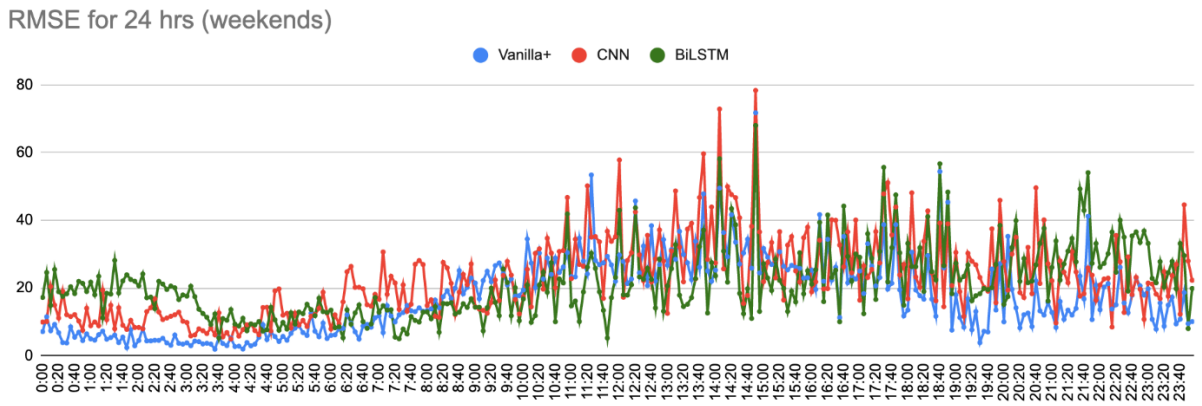

Figure 47 RMSE for 24 hours (weekends)

As shown in Figures 46 and 47, we can see the error patterns for 24 hours at weekends. Compared with weekday data, the weekend errors are much larger and more chaotic. The highest error peak is around weekend afternoon peak hours (14:40). However, the high error lasts until midnight.

In short, it's easier to impute an accurate traffic flow value on weekdays compared with the weekend. And the higher traffic flow may cause a higher error.

CHAPTER 6: Experimental results after data separation

Section 5 shows the different error patterns for weekdays and weekends, and it shows that the error for data of weekends is much higher than on weekdays. In this section, the whole dataset is separated into weekdays and weekends, and they are trained, tested, and errors calculated respectively. To compare the effect of separation, all these three groups use the same hyper-parameter without much tuning. The data volume of training is 7200 and 2592 for testing. The details about the period are shown as follows.

	Training period	Training set volume	Testing period	Testing set volume
Total	Feb 1 – Feb 26	7200	Mar 1 – Mar 10	2592
Split-weekdays	Feb 1 – Mar 5	7200	Apr 1 – Apr 14	2592
Split-weekends	Feb 1 - May 1	7200	May 1 – Jun 1	2592

	RMSE			MAE		
Model	Vanilla	Bi-LSTM	CNN	Vanilla	Bi-LSTM	CNN
Total	63.6	64.16	80.21	24.62	27.58	43.12
Split-weekdays	49.12	37.91	50.06	23.02	25.71	37.13
Split-weekends	25.64	22.42	36.73	15.62	13.7	26.57

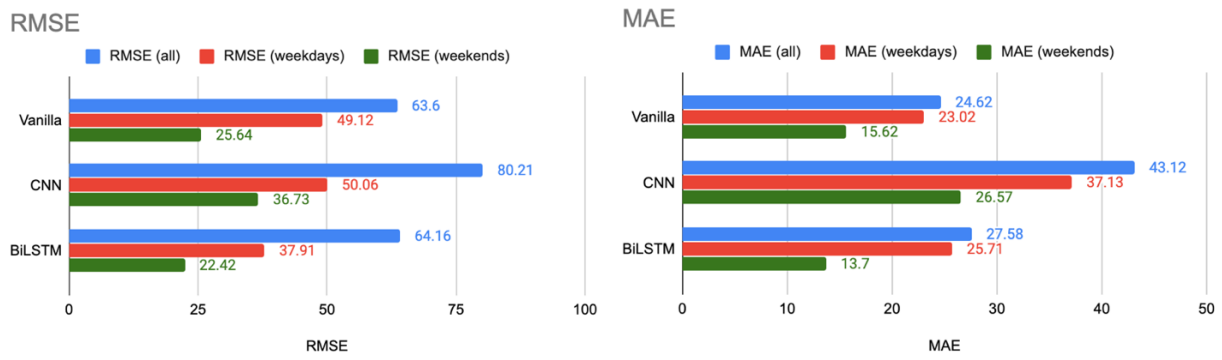

Figure 48 RMSE and MAE after data separation

The testing results are shown in Figure 48. First, the previous conclusion is verified again that the Bi-LSTM is better than “Vanilla” and “Vanilla” is better than CNN. It can seem clear that the training error after splitting the data is much lower than the whole dataset,

especially for the weekend, no matter which model is applied. Therefore, the separation could be a good idea to train the model, and it will offer a more accurate imputation results.

CHAPTER 7: Conclusion

In this section, the results from Chapters 3 and 4 which are the low missing rate part and high missing rate parts are compared, and conclusions from other sections are also summarized.

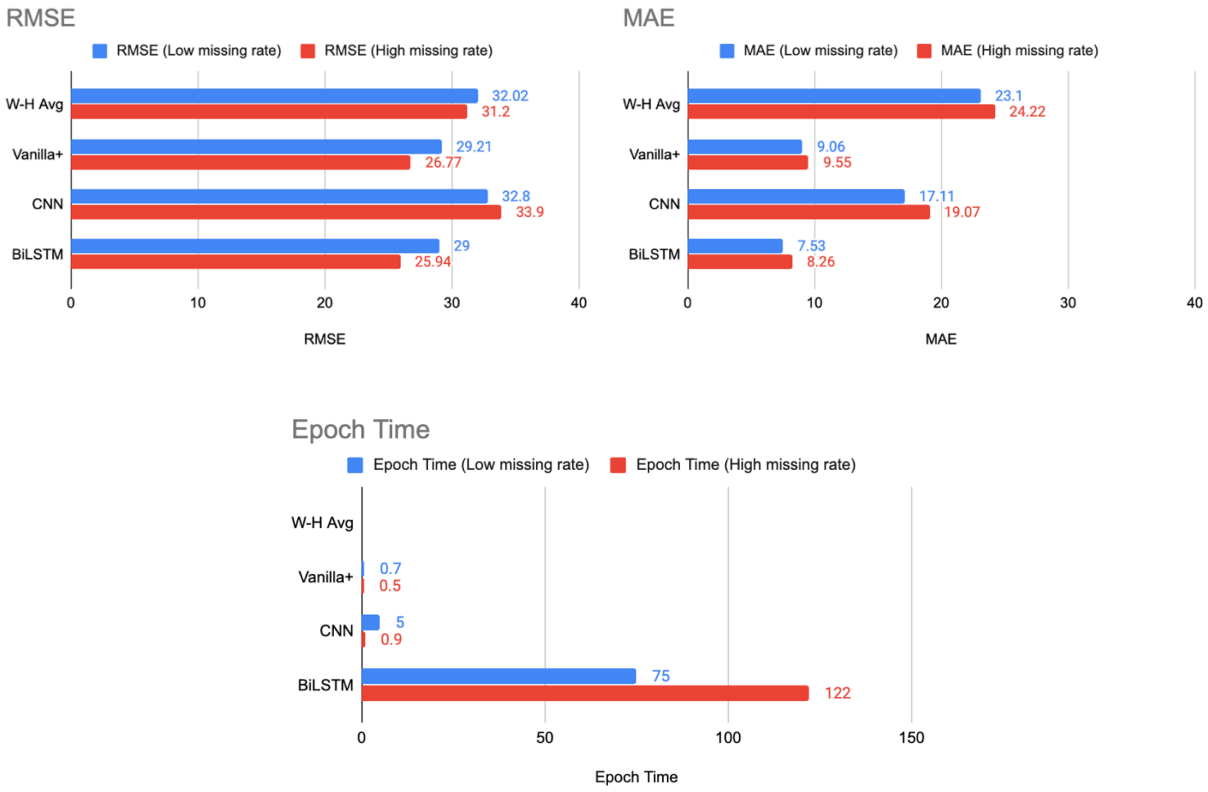

Figure 49 Result comparison of high missing rate and low missing rate

In Figure 49, RMSE, MAE, and epoch times for both high missing rate data and low missing rate data are shown and compared, and three models: "Vanilla", CNN, and Bi-LSTM are also compared to the industry-widely accepted method of using Weekly-Hourly Average of predicting traffic flow.

The training and testing result that Bi-LSTM is better than "Vanilla" and "Vanilla" is better than CNN is shown again. The simplest model "Vanilla" really does very well in the

imputation of missing data even if the missing rate is high. And CNN is not suitable for data imputation though it is very good at image processing. And Bi-LSTM has room for improvement although it's too complex to finish optimization in a short time. Bi-LSTM is much more expensive computationally and yields only slightly better results than Vanilla.

Secondly, the robustness of these three Autoencoders can be verified because the difference of RMSE or MAE between low missing rate data and high missing rate data is small. And the increase of the missing rate doesn't influence the final imputation result much. However, for Bi-LSTM, the high missing rate data has higher computational time and expense. So we can say that these machine learning models can be used for traffic flow data imputation even though the gap within the data is large, for example, the extreme situation case: 80% missing rate.

Besides, two possible reasons for high error problems for one station are proposed and analyzed in Section 4 and it can be studied further after the providing of more detailed corresponding information.

The error patterns for different hours on weekdays and weekends are also shown. For weekdays, the error concentrates at the two daily peak hours in the morning and the afternoon. And the error of weekends is much higher and concentrated from afternoon to midnight.

In the end, the data are separated into weekdays and weekends, trained and tested respectively. And we can see that separation can lower the error significantly, especially for the weekends, and improve the accuracy of imputation.

REFERENCES

[1] G. Atluri, A. Karpatne, and V. Kumar, “Spatio-temporal data mining: A survey of problems and methods,” ACM Computing Surveys (CSUR), vol. 51, no. 4, p. 83, 2018.

[2] B. Bae, H. Kim, H. Lim, Y. Liu, L. D. Han, and P. B. Freeze, “Missing data imputation for traffic flow speed using spatiotemporal cokriging,” Transportation Research Part C: Emerging Technologies, vol. 88, pp. 124–139, 2018.

[3] C. F. Ansley and R. Kohn, “On the estimation of ARIMA models with missing values,” in Time series analysis of irregularly observed data, pp. 9–37, Springer, 1984.

[4] L. Qu, L. Li, Y. Zhang, and J. Hu, “PCA-based missing data imputation for traffic flow volume: A systematical approach,” IEEE Transactions on intelligent transportation systems, vol. 10, no. 3, pp. 512–522, 2009.

[5] R. Asadi and M. Ghatee, “A rule-based decision support system in intelligent hazmat transportation system,” IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 5, pp. 2756– 2764, 2015.

[6] Asadi, Reza. Deep Learning Models for Spatio-Temporal Forecasting and Analysis DISSERTATION. Diss. UNIVERSITY OF CALIFORNIA, IRVINE, 2020.

[7] Costa, Adriana Fonseca, et al. "Missing data imputation via denoising autoencoders: the untold story." International Symposium on Intelligent Data Analysis. Springer, Cham, 2018.

[8] Duan, Yanjie, et al. "A deep learning based approach for traffic data imputation." 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, 2014.

[9] Gondara, Lovedeep, and Ke Wang. "Mida: Multiple imputation using denoising autoencoders." Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham, 2018.

[10] Zhuang, Yifan, Ruimin Ke, and Yinhai Wang. "Innovative method for traffic data imputation based on convolutional neural network." IET Intelligent Transport Systems 13.4 (2018): 605-613.

[11] Zhang, Jianye, and Peng Yin. "Multivariate Time Series Missing Data Imputation Using Recurrent Denoising Autoencoder." 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE, 2019.

[12] R. Asadi, A. Regan. (2019) A convolutional recurrent autoencoder for spatio-temporal missing data imputation, Proceedings of 2019 International Conference on Artificial Intelligence (ICAI'19), pp. 206-212.