MSc Thesis

Extracting insights from black box neural networks for network-wide traffic predictions

Heqi Wang - 5216397

Delft University of Technology

TU Delft

Extracting insights from black box neural networks for network-wide traffic predictions

by

Heqi Wang - 5216397

to obtain the degree Master of Science
in Delft University of Technology,
to be defended publicly on
Feburary 24th, 2023

Chair:	Prof. Dr. Ir. Hans van Lint
Daily Supervisor:	Dr. Ir. Panchamy Krishnakumari
Committee:	Dr. Emir Demirović
	Guopeng Li
Institution:	Delft University of Technology
Place:	Faculty of Transport and Planning, Delft
Project Duration:	April, 2022 - February, 2023

Cover Image: Dutch Highway by Loes Klinker, Unsplash

TU Delft

Preface

Dear reader,

Sitting in front of the computer at the dawn, I feel really excited and cheerful typing these words. After these splendid two-and-a-half years, it finally comes to an end of my adventure as an MSc student majoring in Transport & Planning. What you are reading is my Master thesis, a final milestone manifesting the dedication and effort that I have made in the past months. The topic is on the intersection of traffic prediction and AI explaining, with the attempt to open the black box.

Boldly heading toward this topic without too much experience in AI, at that time I wished to challenge myself. "Why not," I murmured. And now I can confidently say, what a great choice! It's a pure pleasure for me to go through the code grinding, the writing grinding, and learning to critically face the limitations, all of these things. Although I know I wouldn't deliver a perfect thesis because science never ends, the personal growth I have gained is gratifying. The old me could never imagine I ended up obtaining so much knowledge and writing so many lines of code.

Therefore, please let me express my deep gratitude to my committee members first. Hans van Lint, the most charismatic professor I've ever met, thank you very much for being enthusiastic and also rigorous, which helps a lot in shaping my research attitude. Panchamy Krishnakumari, my daily supervisor and nearly a caring friend too, thank you for being with me when I was down, confused, and worried. Being supportive, you always guarantee I have the space to explore by myself as well. Also a big thank you for being my referee during my Ph.D. application, I'll work hard to see you at conferences. The next thanks are to Emir Demirović. As a committee, you provided a lot of useful remarks and feedback, and the guidelines for writing. All of those help me greatly. Finally, Guopeng Li, the person who understands this project the most, besides me, it's you. You supported me a lot with programming and debugging, which facilitates the whole project. I feel honored to have been working and discussing with all of you.

My parents supported me financially and mentally during my study at TU Delft, the most important people in my life, are also my teachers, and my friends. Thank you for bringing me to this beautiful world, raising me, and giving me all your love and care. Cheers to my dearest friends, to the laughing dinners and games, and all the hanging outs, we'll meet at the greatest of accomplishments.

Oh, and cheers to me. One thing I will never regret in this life is studying at TU Delft. I traveled to 10 countries in the past two-and-a-half years, and walked through the mountains and cities; I devoted myself to absorbing knowledge and skills, and eventually determined to pursue an academic career. Thank you for reading.

Heqi Wang - 5216397
Delft, February 2023

Summary

Accurate and trustworthy short-term traffic prediction is crucial in the modern world for the comfort of drivers and decision-makers as it is used to improve the performance of traffic management systems, lessen congestion, increase safety, and shorten journey times. It is possible to discover useful information for network transportation planning, such as forecasting demand, finding bottlenecks, and prioritizing infrastructure improvements, by concentrating on network-wide traffic prediction.

Scholars have developed a variety of methods that can be generally divided into model-based and data-based methods in order to accurately predict network-wide traffic. However, while studies have demonstrated the capability of deep learning methods, particularly convolutional neural networks (CNNs), in predicting traffic states, the complex nonlinear spatial and temporal traffic characteristics, the time-consuming model creation and training, and the unexplained methodology and predictions continue to pose challenges to the task.

This thesis seeks to address these issues by analyzing how deep neural networks identify spatiotemporal traffic patterns for network-wide traffic predictions. To this end, a hybrid CNN-RNN model utilizing a pretrained Inception ResNet v2 feature extractor and a long short-term memory encoder-decoder is constructed to forecast network traffic speeds. A pretrained Inception ResNet v2-based image classifier is built based on the predictions to identify traffic patterns, and Grad-CAM is used to explore how the model identifies them. A freeway network in Amsterdam, Netherlands, is used as a case study.

While it is expected that the hybrid CNN-RNN model can give comparable performance to the state-of-the-art methods, e.g. the DGCN proposed by Li et al., results indicate that it cannot fully capture the dynamic characteristics of the traffic, nor can it accurately provide predictions. The image classifier failed to identify the distinct traffic patterns as well, despite Grad-CAM's success in indicating locations with rapid changes of values.

Overall, the findings highlight the influence of inductive bias on deep learning models, and the importance of fine-tuning and model-data compatibility. Although further research is required, the conclusions are still beneficial to make informed decisions when choosing appropriate models for future network-wide traffic speed prediction tasks.

Contents

List of Figures

List of Tables

1 Introduction

In this chapter, the research background is stated in Section 1.1. Some related works are described in Section 1.2. Section 1.3 introduces the research gap regarding the topic. In Section 1.4, following the identification of the research gap, the main and sub research questions are formulated. At last, Section 1.5 gives the outline of this report.

1.1. Background

Traffic prediction refers to the process of forecasting the future state of a traffic system, such as the number of vehicles on a road network, the speed at which they are traveling, and the amount of congestion. For the convenience of travelers and policymakers, accurate and reliable short-term traffic prediction is essential in the modern world. Short-term traffic prediction helps travelers make better travel plans and choices (e.g., Google Map), and it also provides stakeholders with crucial information that can help them make informed decisions related to environmental sustainability, future development, and traffic management. It can also be used to optimize the performance of traffic control systems and to reduce congestion, improve safety, and reduce travel time for drivers.

In contrast to small-scale traffic prediction that concentrates on local traffic conditions, network-wide traffic prediction problems deal with a larger volume of data spread over a more expansive area. Road segments are no longer isolated, allowing for observation of how disruptions at other locations impact traffic conditions at one location. For instance, significant congestion on a freeway could result from a minor scratch accident several kilometers ahead, or it could also be caused by construction on another freeway. By focusing on network-wide traffic prediction problems, there is a possibility to find valuable information for network transportation planning, including forecasting demand, identifying bottlenecks, and prioritizing infrastructure improvements.

But the challenge exists, as traffic has complex characteristics, both spatial and temporal, and also nonlinear. The unpredictability of various factors such as weather, road conditions, traffic incidents, and human behavior makes it hard to precisely anticipate traffic conditions in real time. The variation of traffic conditions at different times of the day, days of the week, and seasons also creates difficulties in developing models that can adapt to these changes. Additionally, traffic conditions can fluctuate greatly in different areas within a road network, making it hard to make accurate predictions that cover the entire network.

Once a traffic prediction is generated and disseminated for use, it carries with it an inherent responsibility for its potential consequences. Any unforeseeable errors in such predictions may give rise to serious safety concerns. Thus, explainable traffic predictions are especially important, so that stakeholders can more effectively evaluate and compare different models, and identify potential biases or errors. In addition, explainable traffic prediction can facilitate the identification of crucial factors and patterns within traffic data, which might be disregarded by conventional statistical methods, thereby leading to improved accuracy and usefulness of traffic prediction models.

1.2. Related Work

In order to accurately predict network-wide traffic, scholars have introduced various methods for decades, which can be roughly classified as model-based and data-based methods.

Starting from mathematics and statistics, model-based methods are conventional computational approaches studied by researchers, which usually require driver behavior parameters and OD matrices for good simulation. By making use of such tractable variables, model-based methods have an obvious advantage: they provide explainable solutions and reveal traffic patterns. Besides, they are especially useful when planning future projects or modeling demand-expected events. Examples of traffic prediction models include DynaMIT (Ben-Akiva et al., 2000), and VISSIM (Traffic Simulation Software | PTV Vissim | PTV Group, 1992). These software packages allow for the creation and combination of various components, enabling the construction of a network that ranges from a single junction to a complex and realistic large-scale network. Additionally, regulations and policies can be incorporated into the simulation. This makes model-based approaches particularly useful for forecasting and comparing future scenarios.

But the variables needed in the model-based methods often require much effort to obtain, and they are sensitive to data errors. This kind of approach highly relies on the quality and amount of data for the calibration of proper inputs and parameters, which seems not even a problem nowadays in such a big data context. However, even if a large amount of data is on hand, usually the information contained is not sufficient for large networks. Thus, when it comes to large-scale networks, model-based methods sometimes give underdetermined solutions and bring ill-posed problems (Krishnakumari et al., 2018).

The application of data-based methods was born from the willingness of scholars to leverage the tremendous data sources generated from all kinds of infrastructure and devices. Not only do traditional radars, detective loops, and cameras provide us with traffic data, but also smartphones, onboard GPS, and automatic fare collection systems significantly contribute. With the accessible data easily obtained, it seems logical to notice the increasing popularity of data-based methods in practical applications. Examples include support vector regression (SVR), linear regression, and many artificial neural network (ANN) models. Among ANN models, as deep neural networks (DNNs) have more layers between input and output layers, scholars believe that features from more different aspects can be captured and fused together (Liu et al., 2018); and the prediction accuracy of DNNs could be higher (H. Yu et al., 2017). Based on these positives, there has been an increase in the number of data-driven traffic prediction studies utilizing deep learning in recent years.

Despite their benefits, many data-based traffic prediction methods have serious limitations. One of these flaws is the neglect of the spatial characteristics and network structure of traffic data. This lack of physical understanding, which is provided by model-based approaches, makes

it challenging to comprehend the mechanisms driving traffic behavior and the effects of various control strategies. These methods can also be limited in their capacity to reflect the underlying traffic dynamics, such as the influence of road geometry, traffic signals, and other factors on traffic flow. This includes well-known techniques like "Auto-Regressive Integrated Moving Average" (ARIMA) (Williams and Hoel, 2003) and more advanced methods like "LSTM-based RNN" (Yeon et al., 2019).

Academics have begun investigating the potential of transfer learning in addressing traffic issues (Rosario et al., 2018). This is because data-driven traffic prediction methods based on deep learning offer high accuracy and efficient use of data, but require a substantial amount of time and resources for data preparation and model training. Transfer learning is a machine learning technique where a model developed for one task is utilized as the starting point for a model on a different task, thus reducing the time needed for model training (Fang et al., 2015). Pre-trained models are commonly used as the basis for deep learning tasks in computer vision and natural language processing, as they significantly reduce the time and computing resources required to build neural network models from scratch, and can also greatly improve performance on related problems. If transfer learning can be successfully applied to traffic prediction, it could greatly simplify the process of constructing and training suitable models.

Although most works in the traffic speed prediction domain paid no attention to the explainability of their prediction models, a few scholars attempted to employ explainable methods recently, e.g., by utilizing attention mechanisms (Sohn, 2020). Based on the attention weights obtained from their attention-based multi-encoder-decoder model, Abdelraouf et al. extended the explainability by visualizing and interpreting them (Abdelraouf et al., 2022). In a paper related to the early anticipation of traffic accidents (Karim et al., 2022), however, several attention-based explaining approaches were evaluated. It would be interesting to have more work on explaining traffic speed prediction models, out of the ethics and reliability consideration.

1.3. Research Gap

As mentioned above, some data-based traffic prediction methods, even some advanced deep learning approaches, neglect the spatial characteristics and network structure of traffic data. For this issue, solutions have been worked out, e.g., to analyze network traffic as images. This point of view is special, as it adopts the knowledge and approaches of the computer vision domain (e.g., convolutional neural networks) and applies them to traffic identification and prediction. Another way is to utilize graph-based neural networks, thus representing the physical understanding of traffic data in the graph structure.

However, although studies have proven the capability of convolutional neural networks (CNNs) when predicting traffic states (Ma et al., 2017; Krishnakumari et al., 2018), it is unknown how this prediction is performed. This leads to another important problem, i.e., they are in general extremely difficult to explain and hence are also called black box methods. People will certainly wonder about the decision basis of algorithms when deciding whether or not to trust them. The current situation in the traffic domain is that a great number of papers only contribute to the development and application of deep learning methods for traffic prediction, without explaining how they work (e.g. Yao et al., 2019; Tian and Chan, 2021).

While scholars in the AI domain have already made great efforts to interpret the various black box deep learning methods (Ribeiro et al., 2016, Lundberg and Lee, 2017, Q. Zhang, Cao, et al., 2018, J. Wang et al., 2021, Maree and Omlin, 2022), to the best of the author's knowledge,

none of them is traffic prediction related. The spatial-temporal characteristics of traffic data and network topology together make the input very complex, also leading to the difficulty of explaining the method and thus its untrustworthiness. To this end, how these deep learning methods identify traffic patterns when predicting network-wide traffic states remains unknown. Therefore, a study is needed to fill this research gap and provide insights into how deep learning models "learn" when tackling traffic prediction problems, so that reliable and explainable research results can contribute to society.

1.4. Research Questions

In this research, the following main question is aimed to be answered: how do deep neural networks identify spatio-temporal traffic patterns for network-wide traffic predictions?

In order to answer the main research question, some sub-questions are formulated. And specific answers will be found gradually in the research process, which are helpful to finally solving the problem.

- What are the state-of-the-art deep learning methods used for network-wide traffic prediction?
- How to build a computer vision-based network-wide traffic prediction model using deep learning methods?
- Which spatio-temporal properties of traffic contribute to identifying distinct traffic patterns using DNN?
- How to relate traffic patterns to the features extracted using DNN?

By answering these questions, it is assumed that ultimately a series of heatmaps highlighting key areas contributing to specific traffic patterns will be obtained from the chosen method (class activation mapping). First, a computer vision-based DNN model will be constructed to predict traffic speed. Following certain steps, the DNN will be revised, linked to the chosen method, and finally unraveled to a certain extent. The prediction results provided by DNN will be analyzed for traffic pattern identification. Unraveled DNN and generated class activation maps can tell us what information the network is actually absorbing, as well as how important different features are for neural networks when learning traffic patterns. As a result, explainable DNN predictions could be more reliable and persuasive, thus promoting the application of DNN in all kinds of fields, including in our case, network-wide traffic congestion identification and prediction.

1.5. Outline

The rest of the thesis report is organized as follows: Chapter 2 is a literature review regarding related works in deep learning, traffic prediction methods, and class activation maps. The methodology used in this research is described in detail in Chapter 3, which consists of 4 stages. In Chapter 4 a case study performed in the project focusing on a freeway network in Amsterdam is introduced. The experimental setup is also recorded in the same chapter. Results and conclusion are respectively presented in Chapter 5 and 6.

2

Literature Overview

This study incorporates and connects two fields: traffic speed prediction using data-driven methods and the explanation of black box methods like DNN. Considering the model complexity of deep learning methods, the literature overview will be divided into 3 parts, corresponding to the two fields and an introduction to DNNs. The first section reviews the basic concepts and classic models of the deep learning domain. Different types of data-based traffic prediction problems and related works are described in Section 2.2. And in the third section, the efforts of explaining black box methods are presented. Finally, in the last section, a summary of the literature overview is given, and the research scope is determined.

2.1. Deep Neural Networks

Deep neural networks (DNNs), true to the name, have deep structures. It means that these neural networks are built by stacking artificial neural network layers, thus being multiple layers deep. So, as a framework for deep learning, a DNN is a neural network with at least one hidden layer. Similar to shallow neural networks, DNNs can also model complex nonlinear systems. But the extra hidden layers give DNNs the possibility of extracting, learning, and relating more features, thus improving their ability to solve sophisticated real-world problems (Sze et al., 2017). The simplest component of neural networks is an artificial neuron, whose structure is shown in Figure 2.1.

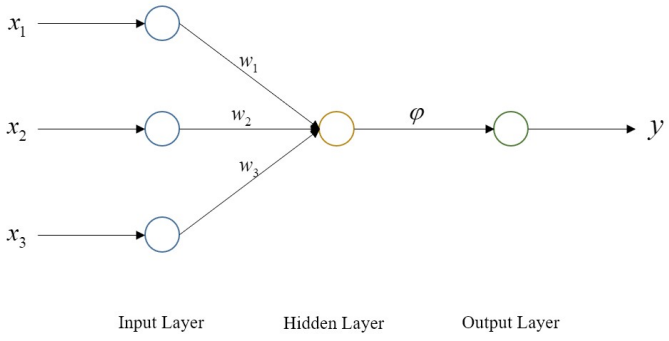


Figure 2.1: Artificial neuron structure

The mathematical expression of the artificial neuron is:

$$y = \phi(\sum_{k=1}^{K} w_k x_k + b) \tag{2.1}$$

where ϕ is the activation function, w_k is a vector of weights, x_k is a vector of inputs, and b is a bias of the neuron. An artificial neuron learns the values of weights w_k during the training process.

The simplest one among the most common activation functions for artificial neurons is Binary Step function, with the formula

$$f(x) = \begin{cases} 0, & x < \theta \\ 1, & x \geq \theta \end{cases} \tag{2.2}$$

The output y of this activation function depends on whether the input meets a specified threshold θ. This activation function can be used in binary classifications; however, it cannot be used in situations where multiple classes need to be dealt with. Therefore, to adapt to different scenarios, other common activation functions like Sigmoid, tanh, Rectified Linear Unit (ReLU), are widely used in neural networks. By using different activation functions, the expressive power of neural networks is further enhanced.

When it comes to artificial neural networks (ANN), multiple artificial neurons are utilized in one model. A small example of a multilayer perceptron (MLP, a full-connected neural network) structure can be found in Figure 2.2. In fully-connected neural networks, each neuron is connected to all neurons on its adjacent layer(s). Although not very "deep" as it seems, MLP is a DNN, as 2 hidden layers are involved in its structure.

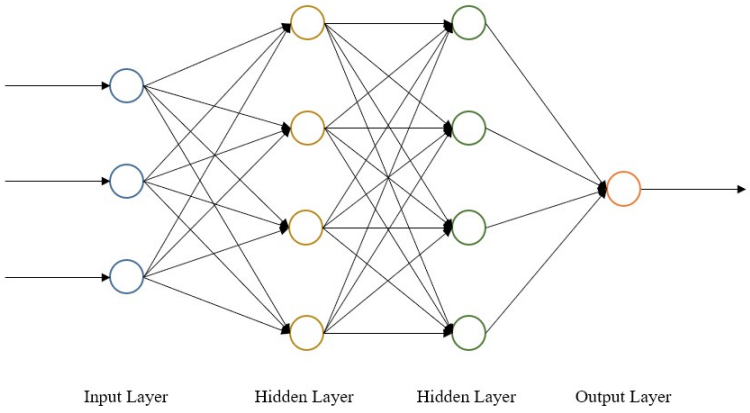


Figure 2.2: An example of full-connected neural network structure

The number of parameters in a MLP is

$$i \times h_1 + \sum_{j=1}^{n-1} (h_j + h_{j+1}) + h_n \times o + \sum_{j=1}^{n} h_j + o \tag{2.3}$$

where the number of hidden layers is n, and h_k is the number of neurons in k-th layer.

Hence, if we keep deepening our network layers and increasing the number of neurons in each network layer, numerous parameters will be needed, thus the model being complex. The more

sophisticated the model, the more difficult it is to modify the parameters and the easier it is to overfit. Additionally, we can observe from the neural network's back-propagation process that, when a gradient is back-propagating, continuous iterations will result in the gradient getting smaller and smaller until it eventually vanishes. With the aforementioned contents as the basis, a number of variants have evolved. In this research, the focus is on convolutional neural networks (CNN), recurrent neural networks (RNN), and graph neural networks (GNN).

2.1.1. Convolutional Neural Networks

Convolutional neural networks (CNN) are a subclass of feed-forward neural networks with a deep structure and convolution calculations. It is one of the representative algorithms for deep learning.

Traditional neural networks are not appropriate for use in domains that deal with images. When a fully-connected network topology is employed, too many parameters are needed, and the amount of calculation needed for random back-propagation is enormous. It is not advised to employ typical neural networks in terms of computer power and parameter tuning. CNN mimics the biological mechanism of vision and is capable of both supervised and unsupervised learning. The sharing of convolution kernel parameters in the hidden layer, as well as the sparsity of inter-layer connections, allow CNN to learn grid-like topological features such as pixels and audio with little calculation. As this thesis focuses on the utilization of image data containing the traffic network topology, CNN becomes a reliable choice.

In Figure 2.3, a CNN structure similar to one of the earliest CNNs, LeNet-5, (Lecun et al., 1998) is presented. LeNet-5 was built in the 1980s and applied to recognize handwritten numbers, and from then on more and more complex models were built and trained on larger and larger datasets.

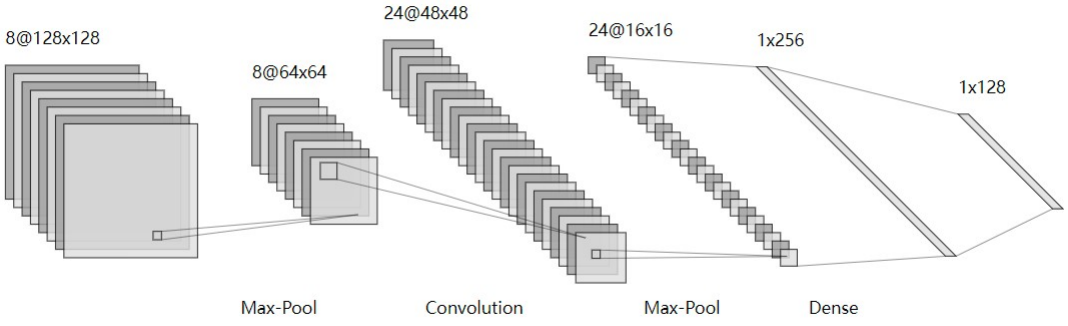


Figure 2.3: A LeNet-like convolutional neural network

Except for the input and output layers, which exist in all artificial neural networks, CNNs have a difference in terms of their hidden layers. The hidden layers of CNNs usually contain convolutional layers, pooling layers, and fully connected layers.

- Convolutional layer

  The input is convolved with a specified number of kernels (filters) in a convolutional layer. The convolution between the kernel and the area of the data that overlaps with it is calculated as a kernel k slides along each point of the input image. A new convolved

image, known as a feature map, is produced as a result of the convolutions performed by the selected kernel across all of the input data. The convolution process converts the data into a set of feature maps by selecting multiple kernels, each with a different weight (one per filter). Since each kernel produces a feature map itself, they all identify and extract a particular pattern from the input. The convolved image of the convolutional layer is then created and output by joining all of the feature maps. The operation process can be found in Figure 2.4.

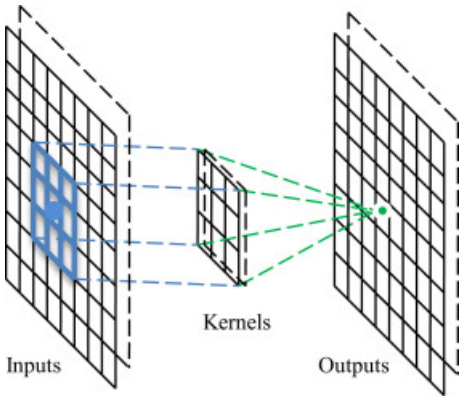


Figure 2.4: Convolutional layer operation, from (Y. Guo et al., 2016)

- Pooling layer

  After convolution operation, the original input would become a new feature map, and usually be passed to the pooling layer. The goal of a pooling layer is to condense the convolved data and decrease the number of parameters without reducing the quality. There are a lot of types of pooling layers, but the most commonly used one is max pooling layer. Figure 2.5 shows the way that a max pooling layer operates by downsampling. It can be observed that there's also a kernel, which moves through the feature maps. At each location, the maximum value from the feature map and kernel's overlapped regions is retrieved, and the output image is constructed using these maximum values. If an average pooling layer is used, then the output image will be constructed using the average values. In this way, pooling layers can minimize the size of the feature maps so that fewer weights are waiting to be learned. These layers produce smaller feature maps, but each of those elements contains data about its neighbors from the preceding layer.

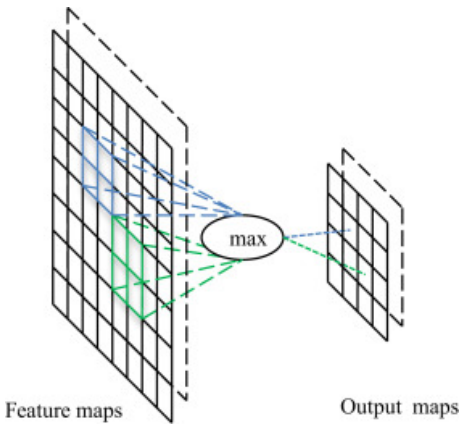


Figure 2.5: Pooling layer operation, from (Y. Guo et al., 2016)

- Full-connected layer

Fully connected layers connect every neuron in one layer to every neuron in another layer, the same as MLP as mentioned before. The feature map will be "flattened" by being thrown into the Flatten layer after the max pooling procedure has been completed. The classification of the images occurs after the flattened matrix passes through a fully connected layer. An example of full-connected layer operation can be seen in Figure 2.6.

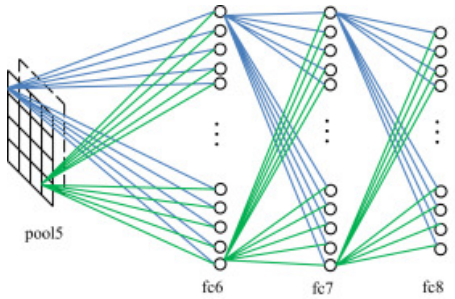


Figure 2.6: Full-connected layer operation, from (Y. Guo et al., 2016)

In some modern algorithm architectures, CNNs might have Inception modules (e.g. Inception v1 by Szegedy et al., 2014) which allow the use of kernels of different sizes in one layer, or residual blocks (e.g. ResNet by He et al., 2015) which avoid the occurrence of the gradient vanishing problem. With the consideration to utilizing them both and improving the models, Inception-ResNet was developed (Szegedy et al., 2016). In this research, Inception-ResNet v2 is adopted for the feature extraction of grid-like speed image data considering its good performance (which achieves 95.1% top-5 accuracy on ImageNet) and transfer learning context.

2.1.2. Recurrent Neural Networks

ANNs can only take and process one input separately, and the previous input has nothing to do with the next input. However, some tasks need to be able to better process sequence information, that is, the previous input is related to the subsequent input. For example, when understanding the meaning of a sentence, it is not enough to understand each word of the sentence in isolation, the entire sequence of these words needs to be processed. When processing video, it is also unreasonable to analyze each frame individually rather than the entire sequence of connections of these frames. When it comes to dealing with speed data, the temporal characteristic of traffic makes it necessary to consider carefully the proper method. Therefore, RNN is adopted as the congestion prediction model for its unique advantage of handling series data.

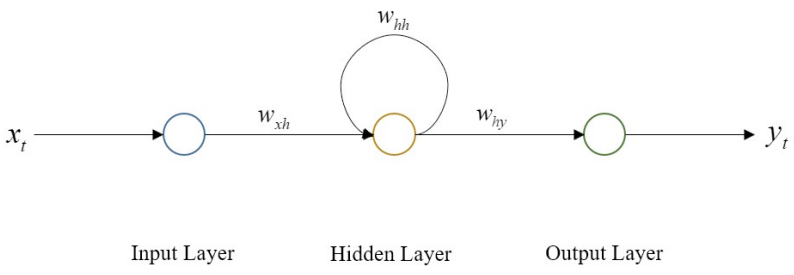


Figure 2.7: A basic RNN structure

RNNs were proposed according to the view that human cognition is based on past experience and memory. One of the earliest works that introduced the concept of recurrent networks is (Rumelhart et al., 1985). The key of RNN to realize the so-called "memorizing" is its structure, which is shown in Figure 2.7 as an example of a fully recurrent neural network. The hidden-to-hidden recurrent connection is represented by the arc, with the corresponding weights labeled as w_{hh}. Input-to-hidden connection is parameterized by weights w_{xh}, and hidden-to-output connection by weights w_{hy}.

Unrolling the RNN by time step is to expand the recurrent kernel in the direction of the time axis (see Figure 2.8). The memory status information h_t is updated at each moment, and the parameter matrices w_{xh}, w_{hh}, and w_{hy} are fixed. It is these parameter matrices that are to be trained and optimized. After the training is completed, the parameter matrix with the best effect is used to perform forward propagation, and the prediction result is obtained. The concept of RNN is to use the recurrent kernel to extract the time features and send them to the fully connected network to realize the prediction of continuous data.

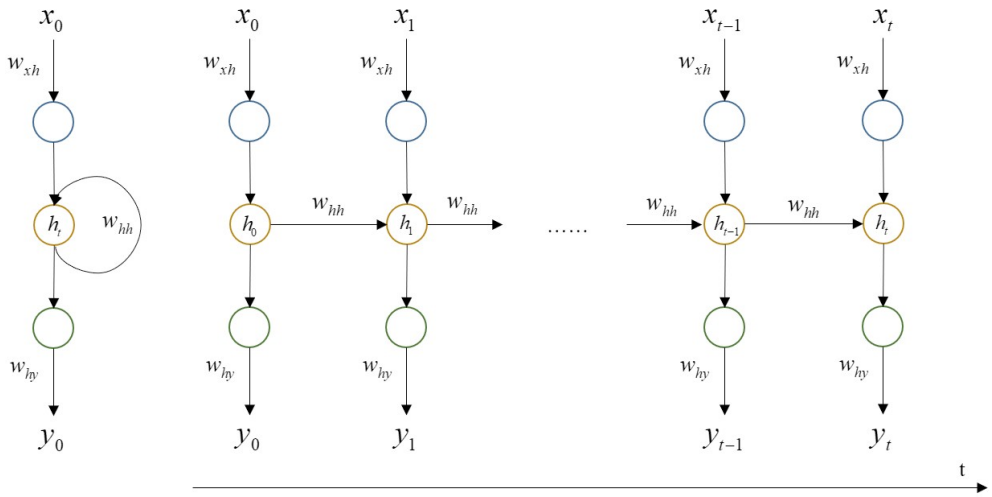


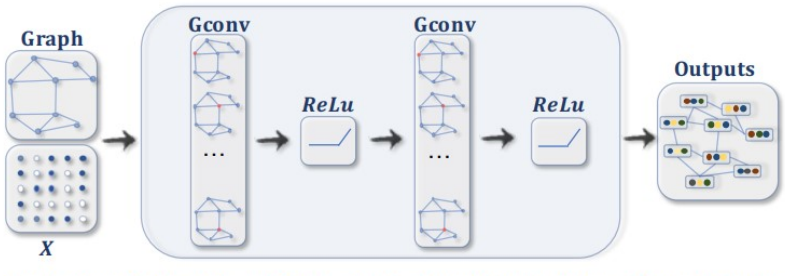
Figure 2.8: Unfolded basic RNN

2.1.3. Graph Neural Networks

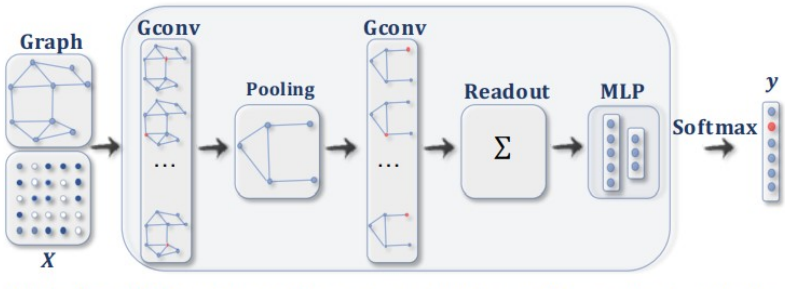
Deep learning methods represented by various end-to-end deep learning paradigms such as CNN and LSTM have achieved great success in extracting features from data in Euclidean space. With that being said, numerous application scenarios produce data from non-Euclidean space, for instance, social networks, biological networks, and transportation networks. Compared with simple text and images, these unstructured data of network type are very complex, and the difficulties in processing them include:

1. The size of the graph is arbitrary, the topology of the graph is complex, and there is no spatial locality like an image.
2. The graph does not have a fixed order of nodes, or a reference node.
3. Graphs are often dynamic and contain multimodal features.

To better model these unstructured data, graph neural network (GNN) was born and developed, inspired by existing deep learning paradigms. GNNs are categorized into 4 classes, which are



(a) A ConvGNN for node classification, from (Rumelhart et al., 1985)



(b) A ConvGNN for graph classification, from (Rumelhart et al., 1985)

Figure 2.9: Example structures of convolutional graph neural networks

respectively convolutional graph neural networks (Conv-GNNs), recurrent graph neural networks (RecGNNs), spatial-temporal graph neural networks (STGNNs), and graph autoencoders (GAEs) (Rumelhart et al., 1985). The structures of each class can be seen in Figure 2.9. GNNs are also becoming more popular in solving traffic-related problems, specifically traffic prediction (B. Yu et al., 2018, S. Guo et al., 2019, Li et al., 2021). To compare which method has higher accuracy, the dynamic graph convolution network proposed by Li et al. (Li et al., 2021) is selected as one of the benchmark models, and will be further introduced in Section 3.6.

2.2. Traffic Prediction

Model-based traffic prediction approaches aim to mathematically model the transportation network and simulate the behaviors of traffic participants. For the purpose of being as realistic as possible, the network structure and relevant traffic information like signal control, speed limit, etc. are usually preserved. Based on these characteristics, model-based methods are suitable for long-term traffic prediction as they can provide insights into the future without actual construction or planning. Examples of developed models include DynaMIT (Ben-Akiva et al., 2000) and TransCAD (Caliper, 1996).

Nevertheless, model-based methods cannot provide accurate predictions, because they're limited by the predetermined network structure and parameter. Although relatively advanced systems were developed to involve various parameters, unexpected events, and external factors are still difficult to capture. The real-time traffic conditions, whose data can be easily obtained and stored nowadays, are thereby excluded in models to a certain extent. To better leverage the data resources and develop more accurate predictions, scholars began to adopt data-based methods for traffic prediction, especially for short-term prediction. To this end, previous work using data-based methods will be introduced in this section in detail.

2.2.1. Different Types of Data-based Traffic Prediction

Given the strong dynamic correlation between the spatial and temporal dimensions of traffic data, it is vital to explore these complex, non-linear patterns to generate reliable traffic predictions. Due to different research focus, a wide range of application tasks are involved in data-based traffic prediction, which can be classified as the following:

- Traffic flow prediction.

  Except for the well-known classic methods HA and ARIMA, conventional learning methods such as support vector regression (SVR) were applied for flow prediction. An early study (Jin et al., 2007) used principal component analysis (PCA) to reduce dimension and recognize temporal and spatial correlations of traffic flow, and then employed SVR to predict based on effective eigen-flows obtained from PCA. Tang et al. improved simple SVR by adding denoising schemes (Tang et al., 2019), and they later further incorporated a fuzzy C-means neural network for more accurate predictions (Tang et al., 2020). Lv et al. were the first to apply a deep architecture neural network model for flow prediction (Lv et al., 2015) by stacking autoencoders as building blocks and learning traffic flow features. Two modules are developed by Guo et al. to encode historical traffic flow data (S. Guo et al., 2019). While the second half employs the graph convolutional network to capture the spatial patterns and common standard convolutions to express the temporal characteristics, the first part uses the attention mechanism to recognize the dynamic spatio-temporal relationships in traffic data.
- Demand prediction.

  In an early work of Kitamura et al.(Kitamura et al., 2000), a micro-simulator was developed and validated to generate daily activity-travel patterns. The modeling of travel demand is quite mature (Domencich and McFadden, 1975, Tajaddini et al., 2020), and well-known approaches for demand prediction include, but are not limited to gravity model (Wilson, 2013, Erlander and Stewart, 1990), and ARIMA (integrated with other methods by Saadallah et al., 2020). Deep learning methods are also widely applied in this problem, for example, Kuang et al. analyzed historical demand data, formed the data into a 4D tensor, and utilized a 3D-CNN to encode external information (Kuang et al., 2019). Similarly, graph neural networks were also adopted out of the transport network topology consideration. Chai et al. incorporated a graph convolutional network and an LSTM network to predict bike flow (Chai et al., 2018), while Yoshida et al. proposed a relational graph convolutional network (R-GCN) for feature extraction of bicycle location (Yoshida et al., 2019).
- Traffic speed prediction.

  Similar to the prediction of flow and demand, general time series prediction approaches (HA and ARIMA) were good choices for speed prediction. With the development of technology, deep learning approaches have usually been selected by scholars in recent years. Ma et al. converted spatio-temporal traffic dynamics to traffic flow images by involving both aspects in a 2D time-space matrix (Ma et al., 2017). Cui et al. proposed a deep stacked bidirectional and unidirectional LSTM (SBU-LSTM) neural network architecture considering forward and backward dependencies in time series data for traffic speed prediction (Cui et al., 2018). Instead of solely using one type of DNN, Wang et al. combined CNN and RNN models by utilizing CNN to learn spatial features and RNN to include periodicity (W. Wang and Li, 2018). A thoughtful study conducted by Liao et al. integrated 3 kinds of auxiliary information, namely offline geographical and social attributes, road intersection information, and online crowd queries, in an encoder-decoder

model (Liao et al., 2018). The attributes and queries were fed to the decoder's fully connected layer, and spatial relations were incorporated with GCN into the encoder model. In this way, the accuracy of traffic speed prediction was improved both during events and over time.

- Travel time prediction.

  In order to choose historical trajectories whose origin and destination locations are comparable to the provided OD input, Wang et al. first employed the kNN technique in terms of OD travel time prediction (H. Wang et al., 2015). They then computed the average value of the selected trajectories as the prediction result. And similarly, the travel time on road segments was predicted by Rahmani et al. using the kNN method (Rahmani et al., 2013). With the goal of considering the temporal dynamic patterns, Zheng et al. learned different weights for time-varying road travel costs in a non-parametric manner (Zheng and Ni, 2013). Yang et al. implemented a spatio-temporal hidden Markov model to incorporate spatio-temporal dependence by dividing the road into a series of adjacent segments (Yang et al., 2013). However, both of their methods only allowed for information sharing between adjacent time slots, failing to take longer-term effects into account. As a result, deep learning methods, such as CNN for local segment spatio-temporal properties and LSTM for the entire route (D. Wang et al., 2018, H. Zhang et al., 2018), began to be utilized.

As mentioned above, due to the tremendous amount of data and some drawbacks of model-based methods, researchers turned to data-based methods, especially machine learning methods. Among all kinds of machine learning methods, neural networks are the most commonly adopted, as they perform automatic feature extraction and prediction in one model. But decades ago, the computational power, theoretical basis, and software limited the training efficiency, so shallow networks were used back then; only no more than ten years ago did scholars start to apply deep neural networks (DNNs) for traffic prediction (Huang et al., 2014).

Different researchers chose different types of DNN for traffic prediction according to their strengths. Recurrent neural networks (RNN) show explicit strength when capturing the temporal characteristics of the data, while convolutional neural networks (CNN) are better at the spatial perspective. Other deep learning-based methods are also being applied, for example, stacked autoencoder (SAE) (Lv et al., 2015) and deep belief network (DBN) (Jia et al., 2016).

2.2.2. RNN Methods

As an advanced method that improves traditional RNN's vanishing gradient problem, the long short-term memory (LSTM) neural network was introduced. And then it became the most popular RNN-based method for traffic prediction, as the compatibility of traffic data with LSTM is high (Tedjopurnomo et al., 2022). One of the earliest works that can be found using LSTM on traffic prediction is from Ma et al. (Ma et al., 2015), in which the speed prediction performance of LSTM was compared with other methods (e.g., SVM, ARIMA, Kalman filter, and so on). The results showed LSTM, at that time, can achieve the best prediction performance, including stability and accuracy.

However, because of the improvements in theory and technology, more complex DNNs could be successfully trained. Thus, a lot of studies combined neural networks and leveraged this hybrid setting for better prediction performance. Examples include a flow prediction work combining CNN and LSTM (Y. Wu and Tan, 2016), in which 3 neural networks were used. They used one CNN for spatial features and two LSTMs for short-term and periodic temporal features,

respectively. Another example is to use LSTM to transform one feature representation to another before passing outputs to a max-pooling layer, and in this paper, the authors predicted traffic congestion (Cheng et al., 2018).

Some works, such as Sun et al.'s paper (Sun et al., 2019), presented a comparison of RNN and CNN. And they used the mean speeds calculated every 5 minutes to divide congestion levels, based on the standard made by Chengdu Transportation Department.

2.2.3. CNN Methods

Different from RNN, how CNN captures spatial features of traffic data strongly depends on the type of data (Tedjopurnomo et al., 2022). When predicting traffic flow, most researchers use point data, for which the spatial features are involved by inducing traffic records from different sensors into vectors (Du et al., 2017). For spatial and temporal features, matrices can be adopted (Fouladgar et al., 2017). Tensors are the best choice for combining matrixed data from multiple days as input (Ma et al., 2017). Although point data are easier to use and transform, most of them are highway data because a lot of city roads have no loop detectors installed.

This makes the study of network traffic difficult, so many scholars use trajectory data that covers a larger region. Krishinakumari et al. mapped the city street network to a 2D grid, transformed the travel time data into average speed, and assigned the speed data to the gridded road segments (Krishnakumari et al., 2018). And Yu et al. used a similar approach (H. Yu et al., 2017). By dividing a city into coarse grids and upscaling gridded data, Liang et al. produced fine-grained urban flow distributions from coarse-grained inputs (Liang et al., 2019). These works show highly visualizable predictions, which is also an advantage of CNN.

Besides spatial features, some researchers also applied CNN for temporal feature capture. For example, Ma et al. contended that RNNs require long input sequences, which increase the training time, and used CNN for both spatial and temporal features (Ma et al., 2017).

2.2.4. Other Deep Learning Methods

As mentioned above, some other DNNs are also applied in traffic prediction, such as stacked auto encoder (SAE), deep belief network (DBN), and graph-based methods, especially graph convolutional neural networks. The number of papers using SAE and DBN is not as large as those using RNN or CNN, which is mainly due to their poor performance caused by not being able to explicitly capture spatial or temporal features (Cheng et al., 2018).

As a breakthrough, graph-based methods have grown in popularity in recent years. Yu et al. formulated the speed prediction problem on graphs to fully utilize spatial information and built the model with complete convolutional structures (B. Yu et al., 2018). However, Li et al. argued that the stacked convolutional layers used in the graph convolutional network make the model structure complex. Apart from that, the static structures are not consistent with dynamic spatial correlations. They developed a variant of graph attention networks named dynamic graph convolution and used this module in an RNN network for multi-step speed prediction (Li et al., 2021).

2.3. Explaining Black Box Methods

Different researchers keep working toward the explainability of black box methods, and one of the directions is visualization. A group of scholars at MIT first introduced class activation map (CAM) to visualize the CNNs (Zhou et al., 2016), which refers to the weighted activation maps generated for each image. A CAM for a particular category can visually show the discriminative area used by CNN to identify this category. But it has to be implemented in a specific network structure by replacing the fully connected layer with a global average pooling layer, thus cannot be generalized in application.

Starting from the initial CAM, quite a few variants were developed. Grad-CAM (Selvaraju et al., 2017) enables the class activation mapping for any built CNN-based image classifier by calculating the average gradients of feature maps. The gradients represent the importance of the corresponding feature map to the target category. However, similar to CAM, Grad-CAM also requires the final convolutional layer in CNN for class activation map generation and cannot localize multiple occurrences of the same class. Grad-CAM++ (Chattopadhay et al., 2018) was then proposed to solve the localization problem. With the help of Smilkov et al.'s SMOOTHGRAD (Smilkov et al., 2017), Omeiza et al. improved Grad-CAM++ by visually sharpen gradient-based sensitivity maps (Omeiza et al., 2019). Their Smooth Grad-CAM++ also makes it possible in terms of the visualization of not only feature maps, but also convolutional layers and neurons.

Apart from CAM methods, more approaches were considered or proposed by other scientists. Most of the works focus on post-hoc explanation of networks, as it is intuitive, does not intervene in the model construction/training, and has a wide range of applications (Castro et al., 2002). Recently, the number of works using active explanation has increased (M. Wu et al., 2019; Q. Zhang, Wu, et al., 2018). Bau et al. proposed a method named "Network Dissection" to quantify the interpretability of latent representations of CNNs (Bau et al., 2017). While Zhang et al. chose decision tree to interpret CNNs, mining all potential decision modes of the CNN (Q. Zhang, Yang, et al., 2018). In both post-hoc and active explanation papers, many scholars extracted rules from the networks. Post-hoc examples include contrastive explanations (Dhurandhar et al., 2018) and interpretable partial substitutes (T. Wang, 2019). Decision tree learning is also an example of post-hoc rule extraction methods, such as using CART or C4.5 to fit the inputs and outputs of a network.

2.4. Summary

Traffic is nonlinear and complex, with both spatial and temporal characteristics. Traffic prediction involves the prediction of a series of traffic information, such as traffic flow, travel time, traffic speed, and demand. When scholars refer to traffic congestion prediction, some wish to predict the queue length, some wish to predict traffic speed, and some wish to classify the level of congestion with traffic flow.

Among all data-based traffic prediction methods, deep learning methods, especially CNNs, RNNs, attention-based NNs, and GNNs, are state-of-the-art. They have excellent abilities in automated feature learning, handling massive data, and involving non-linear relationships. CNNs and RNNs are proven to be capable of capturing spatial and temporal characteristics respectively. However, these approaches need massive labeled data which are sometimes expensive and difficult to collect. They are also difficult to understand, thus being untrustworthy. The requirement of large amounts of computational resources limits the generalizability as well.

For making black box models more transparent and understandable to humans, scholars either developed explainable models from scratch, or adopted techniques such as visualization of model activations, saliency maps, and post-hoc explanations generated by interpretable models. Among these approaches, saliency maps (class activation maps) are the most intuitive ones for people to understand, which reach a balance between mathematically reliable and visually comprehensible. However, class activation maps only highlight important regions without giving complete explanations of the model. This method also lacks generalizability as they are specific to a single input and prediction.

In this project, the traffic prediction task is to predict average speeds, due to the higher accessibility and directness of the data. Specifically, the average speed refers to the mean value of the speeds collected by a certain loop detector during a short period, and there will exist various average speeds at different locations of the road network. Based on the literature overview, the goal of this research is determined. The goal is to predict the average speeds, identify distinct traffic patterns by constructing a deep learning-based model incorporating CNN and RNN, and use Grad-CAM to figure out how the model identifies them.

3

Methodology

This chapter gives a description of the research methodology. Section 3.1 conceptually stated the framework of the methodology. The data rasterizing method is stated in section 3.2. With the feature extraction method mentioned in Section 3.3, speed prediction using DNN can be found in Section 3.4. The steps for unraveling the DNN and generating class activation maps are introduced in Section 3.5. In Section 3.6, an introduction to the 2 benchmark models adopted in this research is presented.

3.1. Conceptual Framework

From previous studies, it is clear that classic statistical data-based methods (e.g., ARIMA) cannot properly consider spatial dependency of traffic data. As they were born when datasets were relatively small, classic data-based methods are usually not suitable for the current big data context, because they lack efficiency or even the ability to process large datasets. Conventional machine learning methods seem to be a solution; however, despite their efficiency and feasibility, they typically suffer from high computational costs. Besides, the relatively simple structure and limited non-linearity of conventional machine learning methods prevent them from modeling complicated and dynamic traffic problems. Hence, deep learning methods will be a desirable solution.

In order to better capture the spatial feature and achieve a visualizable solution for a network-level problem, convolutional neural networks are popularly implemented for network-wide traffic prediction. Compared to more advanced graph neural networks, CNN is the base of computer vision methods and is more intuitive. CNN is also well proven to be capable to predict traffic speeds, thus being a suitable method for exploring its explainability. To preserve the spatial information and fully utilize the advantage of CNN, in this research, traffic congestion prediction uses travel speed as the input, with freeway network mapping into a 2D grid and assigning the data to gridded road segments. In this way, each pixel represents a geographical area and is spatially adjacent, which can be regarded as a special kind of graph structure. Because of RNN's unique capability to include temporal dependencies, with CNN as a feature extractor, the extracted features will then be fed to an RNN model to capture the underlying context and periodicity of the data. RNN itself will also be solely considered as a prediction model to provide a comparison with feature extracting CNN-RNN hybrid model and evaluate the role and effects of feature extraction.

With numerous prediction images at hand, it's difficult to analyze them one by one when

combined with complicated dynamic traffic conditions. Therefore, to find out common and distinct traffic patterns in the prediction, a deep classification model based on CNN architecture will be developed to classify the images. And CAM helps people by enhancing the interpretability of complex deep learning models, thus providing a visual understanding of what parts of the images are leading to a certain class. Among all the methods scholars have adopted, although not the most mathematically deep and detailed, CAM has an outstanding advantage: it is very presentational, which means not only professionals but also stakeholders who care about the decision making process can understand it without profound scientific insights and knowledge. With important areas highlighted on the road network, a possible correlation between congestion propagation and the actual locations can be explored. As the images studied in this project are sparse and of low resolution, there's no need to implement methods like Smooth Grad-CAM++ for high-resolution explanations. Thus, Grad-CAM would be adopted to help figure out the key features extracted by DNN, considering its intuitive benefits, and analyze the relationship between the features and spatio-temporal traffic characteristics.

Based on the above motivations, the conceptual framework can be summarized as follows. To deal with a large image dataset and conserve resources without losing any crucial or pertinent information, the feature extraction technique can be helpful. The redundant data in the dataset can be decreased with the aid of feature extraction. As the context of transfer learning is focused, the features of traffic state variable (speed) images are extracted using a pre-trained convolutional neural network. Then, the features of the future speed images are predicted with a recurrent neural network. For this step, an LSTM encoder-decoder is constructed. By converting the predicted features back to images, the predicted speeds are obtained. Accuracy can be evaluated by comparing prediction images with ground truth images at the same time steps and calculating the MAE and RMSE. To evaluate and compare the developed model, 2 benchmark models are adopted, namely a simple RNN and the Dynamic Graph Convolutional Network (DGCN) (Li et al., 2021). The error values of the prediction results are aggregated and plotted to assess the predicting capability of the proposed hybrid DNN model. The flow chart is shown in Figure 3.1.

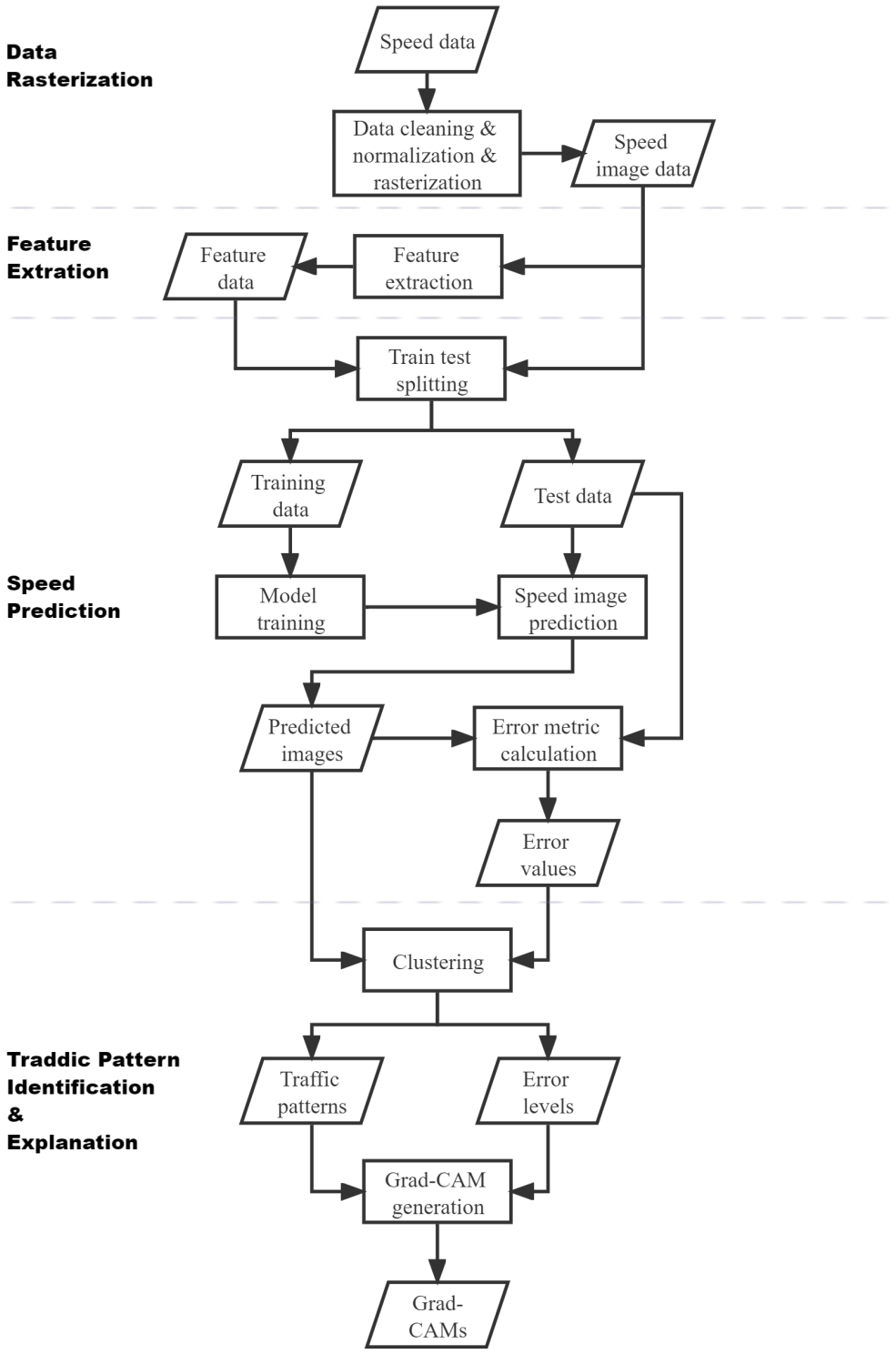
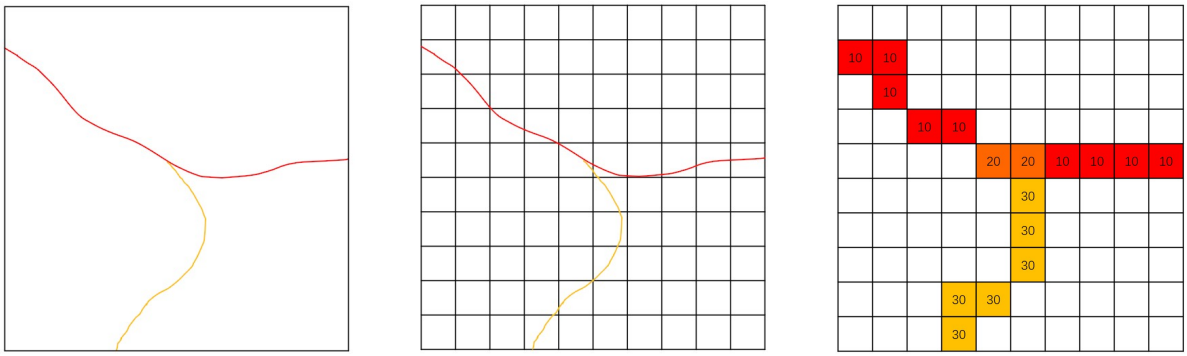


Figure 3.1: Flow chart presenting conceptual framework

3.2. Data Rasterization

By placing loop detectors or cameras along the road, the raw speed data can be collected. In order to preserve the road network's spatial information and utilize deep neural networks, the speed data needs to be "rasterized" so that the input is in image form and maintains the relative positions of loop detectors. One solution to this problem is to encode traffic data in a grid-like matrix, where the grids represent regions of the traffic network (H. Yu et al., 2017). Usually, networks are divided into relatively small resolutions to ensure no links overlap within one grid.

The steps for data rasterization are described below. An example for a small transportation network rasterization can be found in Figure 3.2.



(a) (b) (c)

Figure 3.2: Rasterization example of a small transportation network

Step 1: Select a transportation network with N roads to be studied and cut it evenly into L links with P points. Total number of time steps each day is T, and the number of days is D.

The number of the points P is given by:

$$P = \lfloor \frac{l_n}{dx} \rfloor + 1 \tag{3.1}$$

where l_n is the total length of road n, and dx is the link length. Thus, the number of the links L is given by:

$$L = P - N + 1 \tag{3.2}$$

Step 2: Compute the average speeds at time step t of day d on link n, obtain a speed matrix $\mathbf{V}$ with a size of (D, N, T).

Step 3: Place the transportation network in a square with side length X, and divide the square into small grids of size $(x \times y)$ m^2. As the road network consists of road sections in meters, for the convenience of division, x and y are also length value in meters. It's worth noting that if the network coordinates are latitude-longitude coordinates, an appropriate projection is required to transform the coordinates to meters-meters according to the EPSG code of the study area.

Step 4: Map the average speed data to the grids. With the end points of the links excluded, the other points represent small road segments and are given the speed values measured by loop detectors. For the blank area, the value filled in each grid is 0. For the area with links passing through, the average speed value indicated by the point in the grid is filled

in. If multiple points exist in one grid, then the minimum average speed value is assigned to this grid.

Step 5: The values in all non-blank grids are normalized to [0,1]. Thus, a series of grids with a size of $(\frac{X}{x}, \frac{X}{y})$ containing speed information are obtained. The grid-like image set $\mathbf{I}$ has a size of $(D, T, height, width)$, where $height = \frac{X}{x}$, $width = \frac{X}{y}$.

3.3. Feature Extraction

Since a pioneering work surveyed and classified transfer learning (Pan and Yang, 2009), more and more researchers wish to dig into the ability of transfer learning for numerous models. From recent surveys (Weiss et al., 2016, Zhuang et al., 2020) it can be observed that many studies turned to utilize neural networks in transfer learning, and then deep neural networks. A popular approach for deep transfer learning is utilizing models that are already trained on other datasets (e.g., ImageNet) and removing the last layer(s) (Krishnakumari et al., 2018). As mentioned in section 2.1.1, Inception-ResNet v2 is used as the feature extractor for preprocessed grid-like speed data.

The condensed architecture of Inception-ResNet v2 used in this project is shown in Figure 3.3. After removing the last layer, the grid-like speed images are fed into the model as a tensor of size $(D, T, height, width)$ as written in section 3.2. Then for each time step a feature of size 1536 can be extracted, thus finally a feature matrix of size $(D \times T, 1536)$ is obtained.

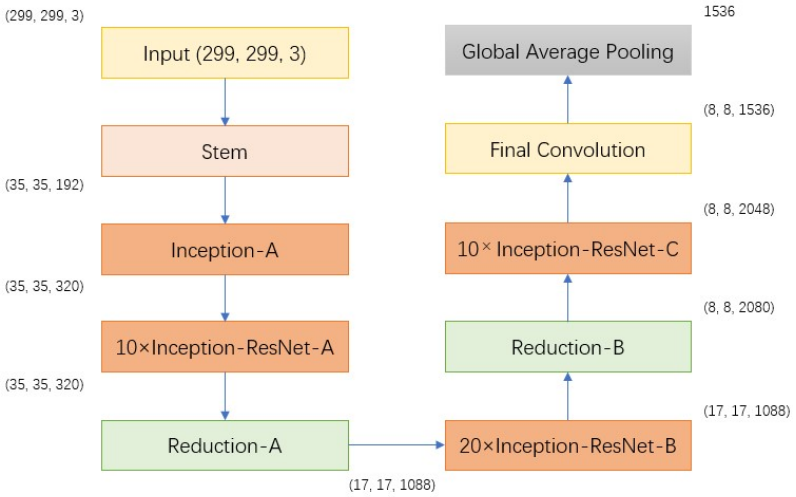
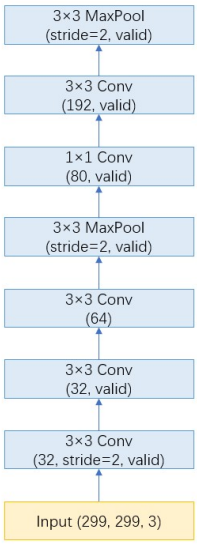
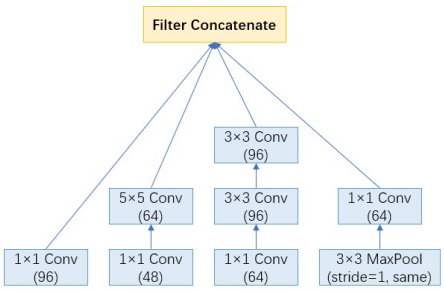


Figure 3.3: Schema for Inception-ResNet v2, self-drawn according to (Szegedy et al., 2016)

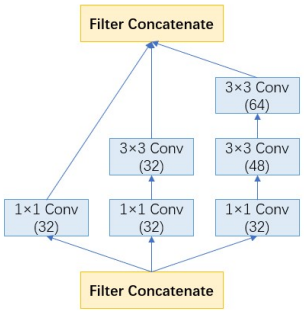
The detailed structures of blocks can be found below in Figure 3.4. The structure adopted in this project is consistent with Keras version Inception ResNet v2 and different from the original paper. The last two layers, i.e., 1 global average pooling layer and 1 dense layer, are excluded as these are used for generating the image classification results of 1001 classes. Weights pretrained on ImageNet are loaded for feature extraction.



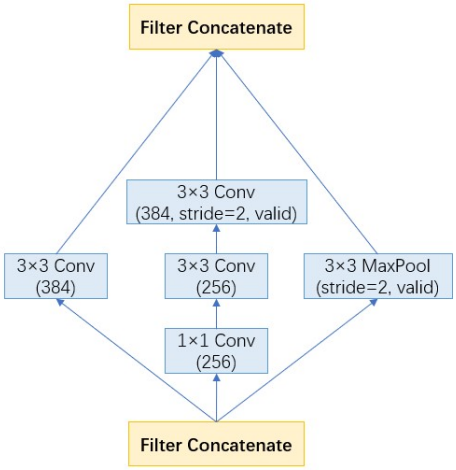
(a) Stem block



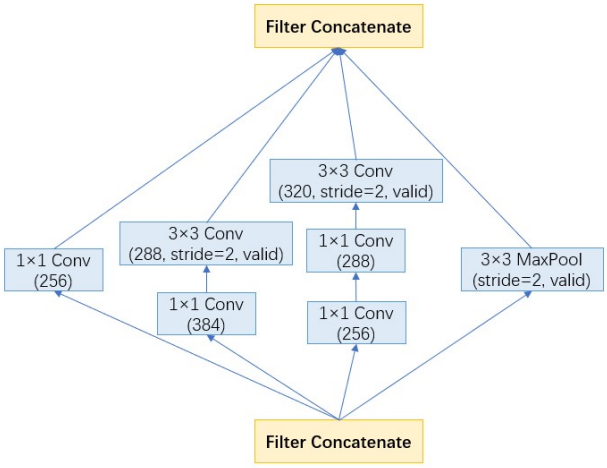
(b) Inception-A block



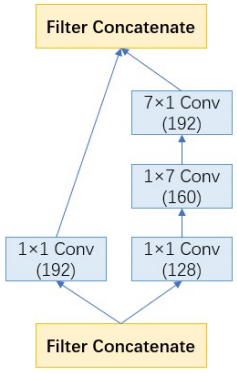
(c) Inception-ResNet-A block



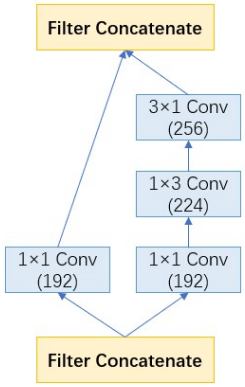
(d) Reduction-A block



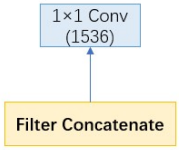
(e) Reduction-B block



(f) Inception-ResNet-B block



(g) Inception-ResNet-C block



(h) Final convolutional block

Figure 3.4: Detailed Inception ResNet v2 feature extractor blocks structure

3.4. Traffic State Prediction

With a series of speed images, the speed prediction problem can be classified as a Sequence2Sequence task. In the field of deep learning, the usual way to deal with these tasks is to utilize RNN encoder-decoder structure, encode the input source sequence into an intermediate context (which can be understood as a code vector of a specific length), and then restore to an output target sequence through the context. Thus, an RNN encoder-decoder consists of 3 parts: encoder, intermediate (encoder) vector and decoder. One RNN is used to simulate the reading action of the brain, one feature vector of a specific length is used to simulate human memory, and then another RNN is used to simulate the action of the brain thinking and getting the answer. The organization and utilization of the three becomes a "simulated brain" that can solve the Sequence2Sequence problems. After the feature extraction, the features are fed to an RNN encoder-decoder for speed prediction (see Figure 3.5).

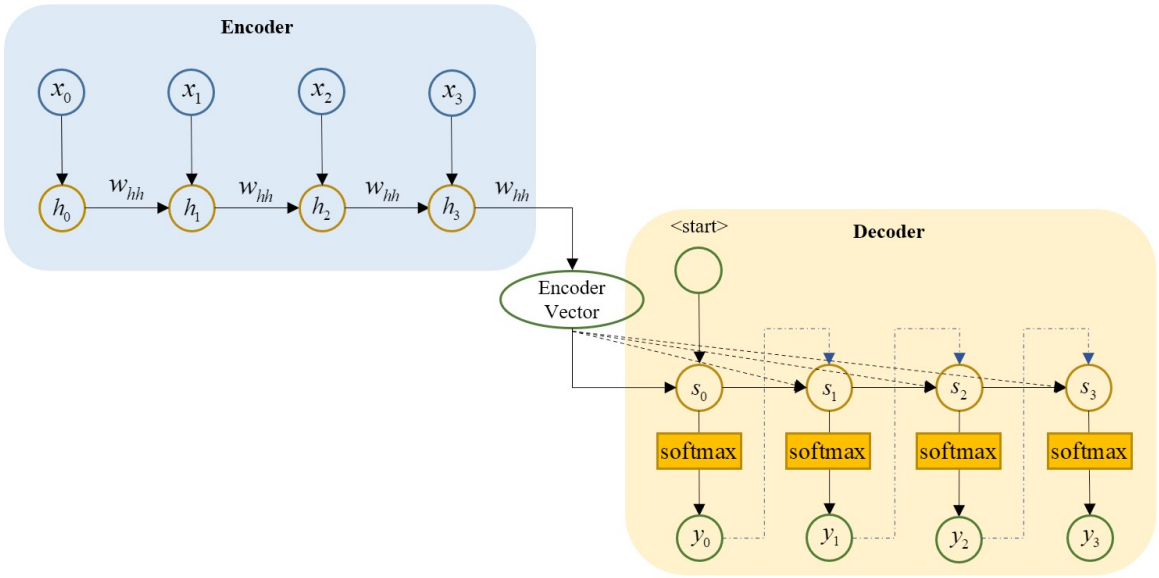


Figure 3.5: An illustration of RNN encoder-decoder

RNN works well in processing time series data, but there are still some problems. A more serious one is that RNNs are prone to gradient vanishing or gradient explosion. In terms of RNN gradient vanishing, here refers to the phenomenon that the memory value becomes small due to too long time period. To tackle this issue, Long Short-Term Memory (LSTM) and other more advanced cells were developed. Different from RNN who only has one state w_{hh}, LSTM has 2 states, namely cell state (which corresponds to the RNN state w_{hh}) and hidden state.

The structure of LSTM cell is shown in Figure 3.6. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. In LSTM cell, the current input at time step t x_t and prior time step's hidden state h_{t-1} are utilized to train 4 states inside the cell (namely, f_t, i_t, $\widetilde{C}_t$, and o_t). With matrices w_* and u_* denoting the weights of the input and recurrent connections, the 4 states are given by:

$$f_t = \sigma(w_f x_t + u_f h_{t-1} + b_f) \tag{3.3}$$

$$i_t = \sigma(w_i x_t + u_i h_{t-1} + b_i) \tag{3.4}$$

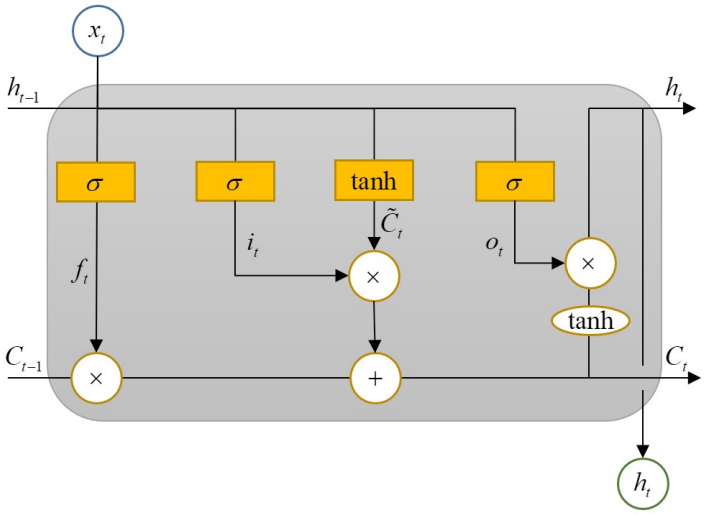


Figure 3.6: LSTM cell structure

$$\widetilde{C}_t = tanh(wx_t + uh_{t-1} + b_c) \tag{3.5}$$

$$o_t = \sigma(w_o x_t + u_o h_{t-1} + b_o) \tag{3.6}$$

where σ and $tanh$ are respectively sigmoid activation function and tanh activation function. f_t, i_t, and o_t are gate activation vectors for forget gate, input gate and output gate respectively, and are in range (0, 1). $\widetilde{C}_t$ is the cell input activation vector in range (-1, 1).

With these states obtained, cell state C_t and hidden state h_t can be inferred:

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{3.7}$$

$$h_t = o_t \odot tanh(C_t) \tag{3.8}$$

where initial values C_0 and h_0 are equal to 0. $\odot$ is the operator for element-wise product (Hadamard Product).

In LSTM encoder, each recurrent unit takes a single input sequence element, extracts information from it, then propagates the information to subsequent elements. The input sequence is a collection of speed images in this thesis. Each input speed image is represented as x_i where i is the time step of that image. Inducted from Equations 3.3, 3.4, 3.5, 3.6, the general encoder hidden states h_i can be given by:

$$h_t = f(w_{xh} x_t + w_{hh} h_{t-1} + b) \tag{3.9}$$

where w_{xh} and w_{hh} are weights for input-to-hidden connections and hidden-to-hidden recurrent connections respectively.

While in LSTM decoder, each recurrent unit receives an input and a hidden state from the preceding unit before producing an output and a hidden state of its own. Each predicted speed image is represented as y_i where i is the time step of that image. Any hidden state h_i is computed by:

$$h_t = f(w_{hh}h_{t-1} + b) \tag{3.10}$$

Then the output y_t at time step t is calculated using the hidden state at that time step and respective weight w_s:

$$y_t = softmax(w_s h_t + b) \tag{3.11}$$

Notably, as the inputs of LSTM encoder-decoder are features extracted by Inception ResNet v2, a processing is needed to convert the predicted features back to images of the same size and form as inputs. Therefore, an operation named transposed convolution is adopted. In a convolution process, take conv2D layer as an example, for an input tensor of size $(N, C_{in}, H_{in}, W_{in})$ and corresponding convolutional layer output of size $(N, C_{out}, H_{out}, W_{out})$, below equations hold:

$$H_{out} = \lfloor \frac{H_{in} + 2 \times p[0] - d[0] \times (k[0] - 1) - 1}{s[0]} \rfloor \tag{3.12}$$

$$W_{out} = \lfloor \frac{W_{in} + 2 \times p[1] - d[1] \times (k[1] - 1) - 1}{s[1]} \rfloor \tag{3.13}$$

where p is padding, d denotes dilation, k is for kernel size, and s for stride.

Now consider conv2DTranspose, with the same form of input $(N, C_{in}, H_{in}, W_{in})$, the following equations can be used to calculate the shape of output tensor, which in our case is the prediction image tensor transformed from features, or determine other parameters with known of input and output:

$$H_{out} = (H_{in} - 1) \times s[0] - 2 \times p_i[0] + d[0] \times (k[0] - 1) + p_o[0] + 1 \tag{3.14}$$

$$W_{out} = (W_{in} - 1) \times s[1] - 2 \times p_i[1] + d[1] \times (k[1] - 1) + p_o[1] + 1 \tag{3.15}$$

where p_i and p_o are padding for input and output respectively.

3.5. Traffic Patterns Identification and Explanation

In order to analyze underlying traffic patterns in the speed prediction images, an image classification model is required. Due to the capability of processing image data and localizing parts in the images, the convolutional layers should be used in the classification model. As Inception networks are famous pretrained models for image classification tasks and also used in this project, it's a reasonable choice to keep it, which can reduce the complexity of codes and avoid stacking various deep learning models. Therefore, by utilizing Inception ResNet v2 structure, an image classifier is built with the following architecture 3.7.

It is clear that only the final dense layer is changed to output the probabilities of 3 classes, instead of initial 1001 classes. By passing the "include_top=False" argument to keras.applications.inception_resnet_v2.InceptionResNetV2, the model keeps its convolutional blocks and drops the last 2 layers (just the same as what has been done in the feature extraction part). Apart from the above mentioned merits, making use of Inception ResNet v2 is also a good opportunity to test the capability of transfer learning strategy under traffic pattern identification context. Hence, to this end, the preserved layers are given weights obtained from model pretraining on ImageNet dataset and frozen to be untrainable. Only the 2 manually added layers are trained with speed images and corresponding class labels. This can save a considerable amount of model training time as well, thus also a suitable approach for organizations

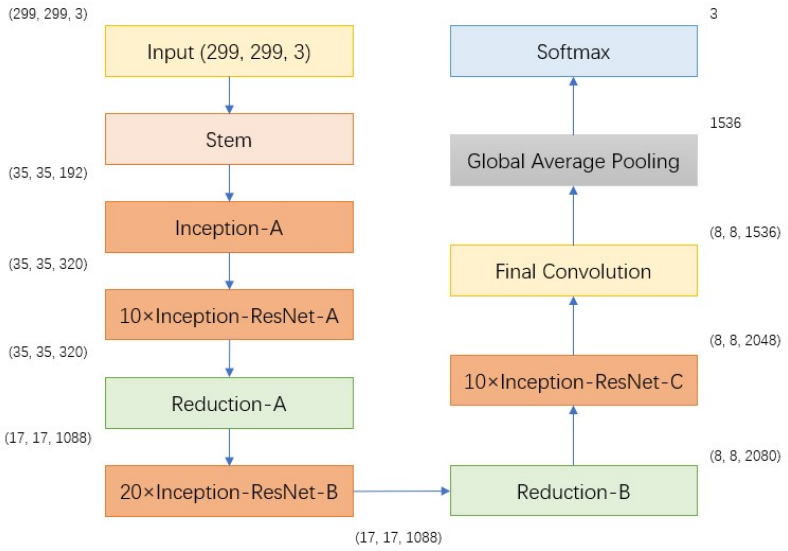


Figure 3.7: Inception ResNet v2 based image classifier

or institutions who have equipment with limited computational power.

For now, as the Inception ResNet v2 is modified to be able to classify the speed prediction images, the weights of the convolutional layers inside the model trained after adding the GAP layer and dense layer can be used to generate class activation maps. The process of traditional class activation mapping is shown in Figure 3.8.

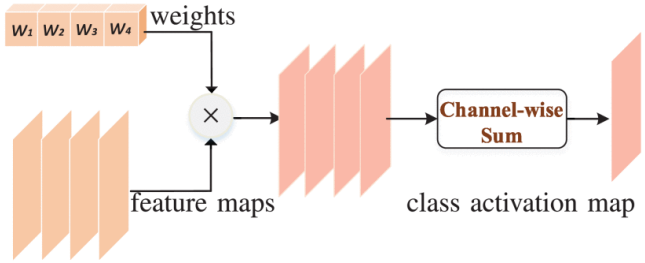


Figure 3.8: Class Activation Mapping process, from (Jiang et al., 2021)

Grad-CAM (Figure 3.9) is a class-discriminative localization technique, which generates visual explanations for any CNN-based network without structural modification or re-training (Selvaraju et al., 2017). From Figure 3.9 we can see that guided Grad-CAM can be generated based on Grad-CAM by guided backpropagation, which shows the fine-grained heatmap of the cat and provides a more detailed visualization. However, as the speed images used in this project are sparse images without contours and shapes, Grad-CAM is sufficient for the goal of finding key areas on the network.

The detailed mechanism of Grad-CAM generating class activation maps is as follows:

Step 1: Calculate the gradient of the score y for class c, with respect to feature map activation A^k of a convolutional layer:

$$\frac{\partial y^c}{\partial A^k_{ij}}$$

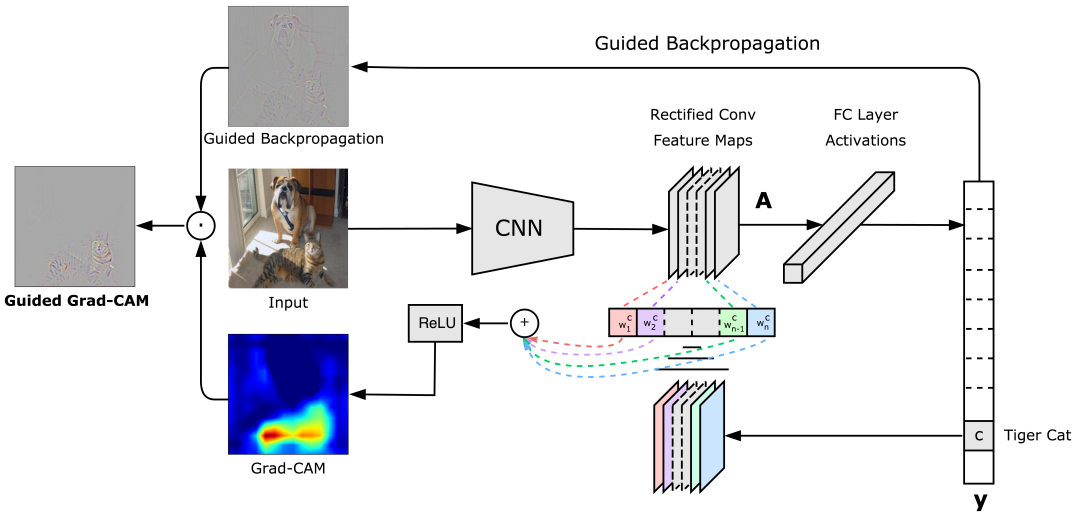


Figure 3.9: Grad-CAM overview, from (Selvaraju et al., 2017)

where i and j denote height and width dimensions respectively.

Step 2: Obtain weights W_k^c by global average pooling the flowing back gradients.

$$W_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{3.16}$$

where W_k^c includes the importance of feature map k for target class c, $Z = i \times j$ is the size of feature map.

Step 3: Perform a weighted combination of forward activation maps and feed it to a ReLU function.

$$L_{Grad-CAM}^c = ReLU(\sum_k W_k^c A^k) \tag{3.17}$$

where $L_{Grad-CAM}^c$ is the generated Grad-CAM for class c, $ReLU$ is rectified linear unit activation function.

The result of above steps is a coarse heatmap of the same size as the convolutional feature maps. Notably, the reason why ReLU is adopted after obtaining the weighted feature map combination is that only features that positively affect the class of interest are focused, i.e., pixels whose intensity should be increased to enhance the probability y^c of class c. And negative values indicating pixels that are likely to belong to other classes are discarded in this way.

3.6. Benchmark Models

2 benchmark models are selected for assessing the prediction performance of the aforementioned hybrid DNN model, which are an RNN encoder-decoder and a dynamic graph convolutional network.

In order to explore whether CNN feature extraction improves the prediction accuracy, the LSTM encoder-decoder is taken out as a benchmark model. The structure remains the same as in Figures 3.5 and 3.6, and the only difference is the model input. Instead of taking the extracted feature of gridded images as input, the images are directly fed to the LSTM encoder-decoder as 4D tensors. By comparing this model to the proposed CNN-RNN hybrid model, the efficiency

and effects of feature extraction on this problem can be assessed.

A dynamic graph convolution model was designed by Li et al. to capture the propagation of congestion (Li et al., 2021). By incorporating an adjacency matrix that encodes the connectedness, DGC is an expansion from Euclidean space to graphs. Thus, it is capable to preserve real location information and extract spatial correlations. The model structure is shown in Figure 3.10. The DGC's input is divided into two tensors, namely feature maps and traffic states. Three components make up a DGC module:

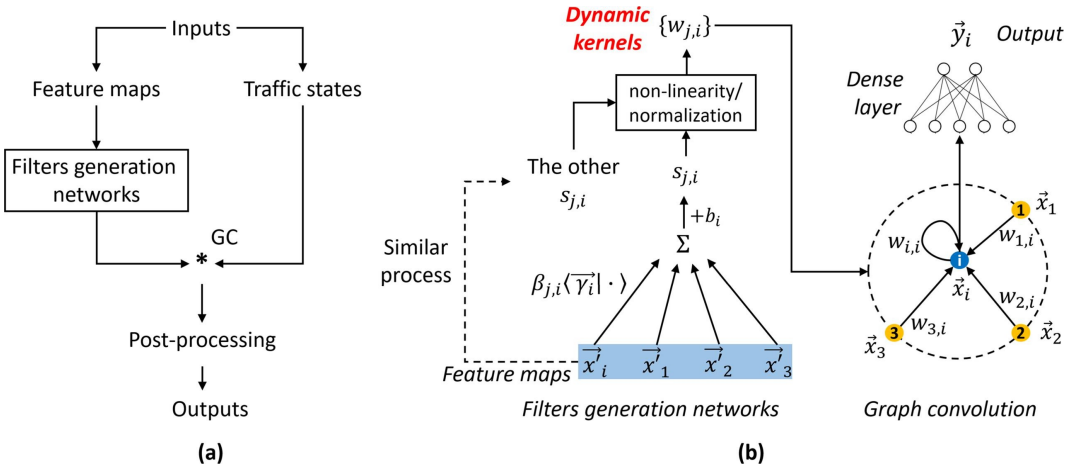


Figure 3.10: DGC module: (a) the structure of a DGC module;
(b) the details of filters generation networks and the DGC graph aggregator. (Li et al., 2021)

1. a filters generation network (FGN), which calculates dynamic graph convolutional kernels using different feature maps;
2. the DGC kernels are subsequently used in a local-wide graph convolution with traffic states;
3. if necessary, post-processing can modify the dimension of the outputs.

Based on the design of DGC modules, multistep traffic forecasting was performed by placing the modules in an RNN encoder-decoder. With the RNN encoder-decoder structure presented in Figure 3.5 applied, the cells are substituted to DGCN cells. Figure 3.11 shows the inner structure of a DGCN cell.

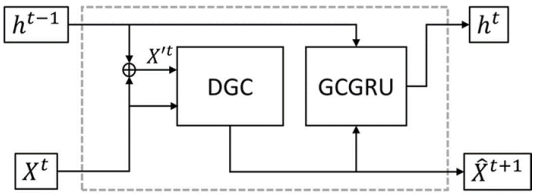


Figure 3.11: DGCN cell structure, from (Li et al., 2021)

By comparing the DGCRNN with the proposed hybrid model, some insights can be gained on whether DGC modules or CNN layers perform better.

4

Case Study

In this research, an urban freeway network in the Netherlands is selected as a case study. It is a relatively complex freeway network around Amsterdam that contains multiple intersections, denoted by "AMSnet". A map of the study area is shown in Figure 4.1.

Due to its wider coverage, AMSnet is consistently divided into 400 m links to minimize complexity and thus has 201 links. Speed data obtained from Li et al. (Li et al., 2021) is the speed of the entire year 2018, and holidays and weekends are excluded due to the lack of congestion. Due to the limitation of computation power, data of 100 days without broad-scale sensor malfunction are selected and prepared. In order to include as diverse traffic patterns as possible, the data during the afternoon and the evening peak lasting for 6 hours is preserved, and the dataset is named AMSnetE9.

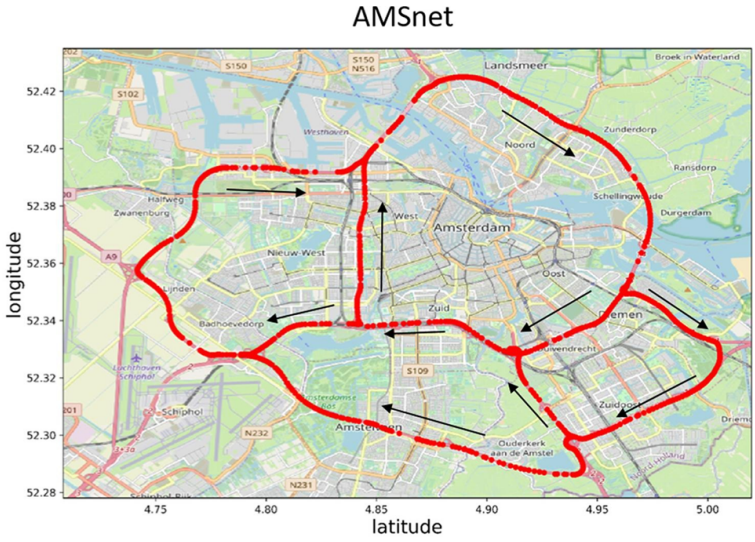


Figure 4.1: Freeway network of Amsterdam with driving direction marked, from (Li et al., 2021)

The traffic speed prediction task in this project is to forecast the speeds of future 10 time steps with a 15-time-step observation window. With the data aggregated every 2 minutes, it is clear that the observation window is 30 minutes and the prediction horizon is 20 minutes. Preprocessed AMSnetE9 is a 3D matrix of shape (100, 180, 201), which includes the speed data of 180 time steps on 201 links in 100 days.

The freeway network is a series of latitude-longitude coordinates, which are transformed to x-y coordinates according to the Netherlands EPSG code. The projection is achieved by transforming EPSG:4326 to EPSG:28992 using pyproj package. Then a series of shapely LineString objects are generated for link partition. Each link is represented by its start point when mapping the speed to the freeway network, indicating the speed collected by sensors on the road. According to the scale of AMSnet, it is placed in a square of $20,000m \times 20,000m$ with meshed grids constructed by 51×51 mesh points. Each grid has a size of 400 m $\times$ 400 m. An example of AMSnet at time step 120 of day 50 can be seen in Figure 4.2.

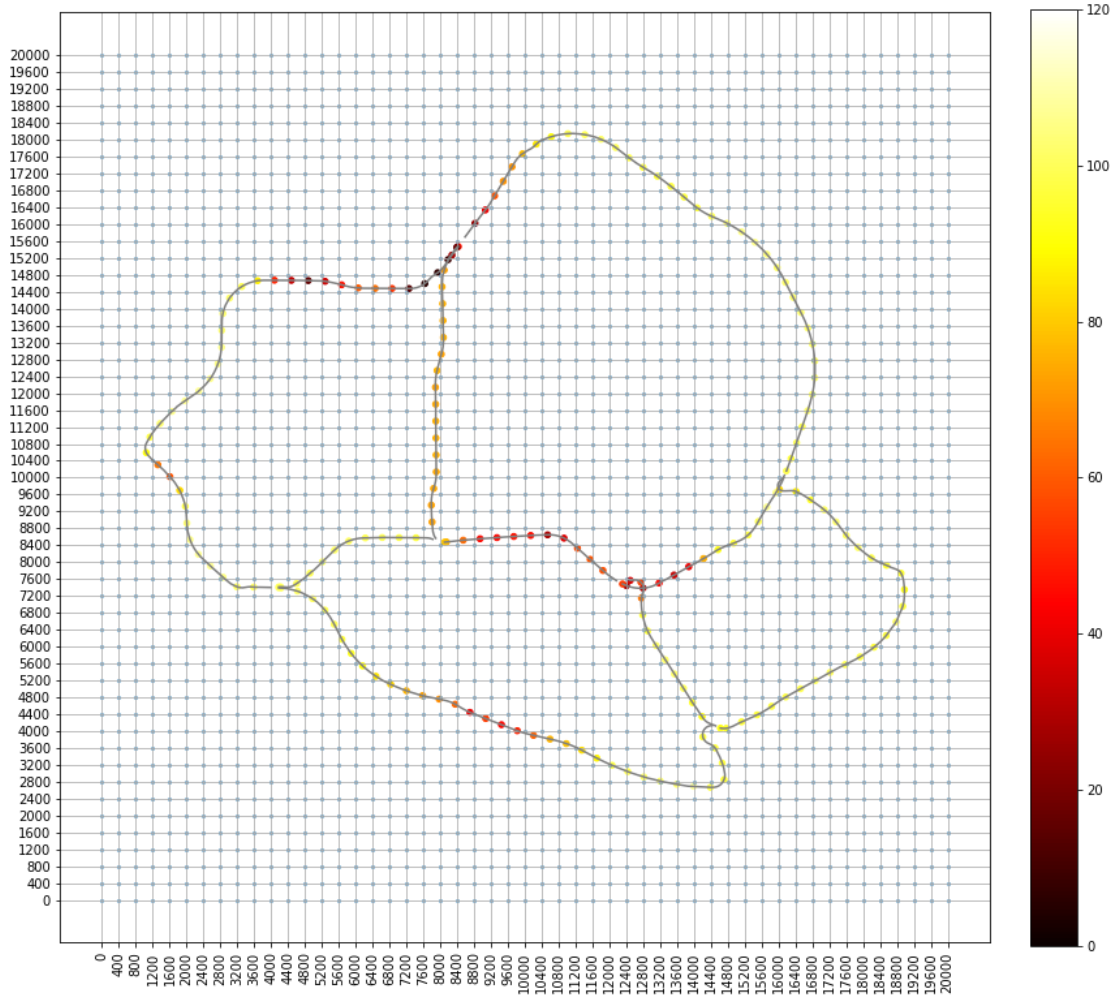


Figure 4.2: Mesh gridded AMSnet at time step 120 of day 50

All colored points in Figure 4.2 represent a link with mapped speed, and are considered to locate inside a grid in the case of:

$$meshx[j,j] \leq ipx[i] < meshx[j+1,j+1] \;\&\&\; meshy[k,k] \leq ipy[i] < meshy[k+1,k+1] \tag{4.1}$$

where *meshx* and *meshy* are respectively the sets of x and y coordinates for mesh points, and *ipx* and *ipy* are respectively the sets of x and y coordinates for colored points.

If a grid has a colored point inside it, it is considered to have a road passing through and is then filled with the corresponding speed value. In the case of multiple colored points located

in one grid, the largest speed value is preserved and filled. Blank grids with no road passing through are assigned with value -999. In this way, the speed data are mapped and rasterized into a 3D matrix of size $(18000, 50, 50)$. Rasterized speed image at time step 120 of day 50 is shown in Figure 4.3, whose size is $(50, 50)$.

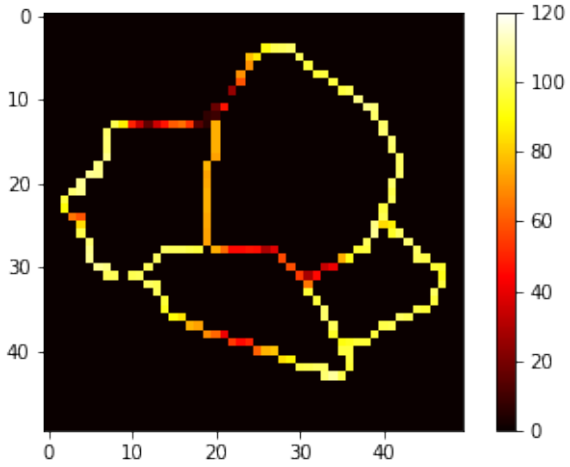


Figure 4.3: Rasterized speed image at time step 120 of day 50

Images are then fed to Inception ResNet v2 feature extractor after they are resized up from size $(18000, 50, 50)$ to $(18000, 299, 299)$. The interpolation method is INTER_NEAREST, for preserving the original network structure as much as possible. Extracted feature matrix of size $(18000, 1536)$ is partitioned into 15 or 10-time-step datasets as the encoder or decoder input for an encoder-decoder model using sliding window method, and further split into training, testing, and validation sets. Initial unprocessed rasterized speed images are used as decoder output during model training.

All speed data is divided by the 120 km/h speed restriction to standardize it between 0 and 1 before being fed to the models. Standardized data are utilized as input for testing, but the prediction results are then fed into a procedure in reverse to produce true-value forecasts. By measuring the distances between predictions and the corresponding ground truth, the errors are estimated. The mean average error (MAE) and the root mean square error (RMSE) are used as error measurements. MAE can measure the overall accuracy of the model predictions, and RMSE provides insights into the uncertainty and variation. MSE is used as the loss function for training and is minimized by optimizer Adam (Kingma and Ba, 2014). The initial learning rate and decay rate are set to 0.001. Batch size is 6, and the number of epochs is 10. Notably, during the calculation of error metrics, both predictions and ground truths are masked to only consider the values on the freeway network. The visualized mask consisting of ”0”s and ”1”s is shown in Figure 4.4.

To evaluate the impact of the CNN feature extractor, an LSTM encoder-decoder mentioned in Section 3.6 is used as one of the benchmark models. And to compare the proposed model to state-of-the-art, the DGCN model developed by Li et al. (Li et al., 2021) is adopted as another benchmark model. Using the same set of parameters, the same error metrics are also calculated for the benchmarks.

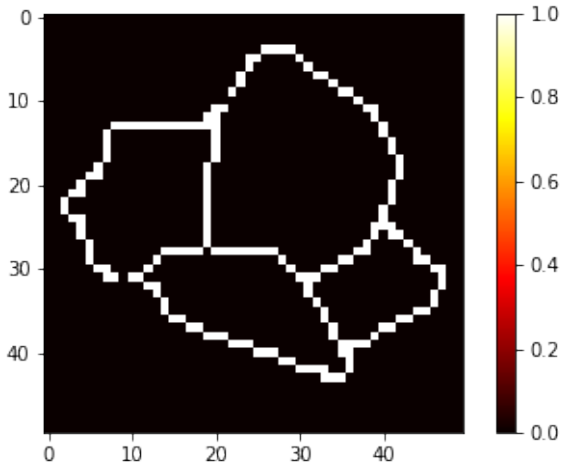


Figure 4.4: Mask

After the speed predictions are obtained, a deep classification model based on Inception ResNet v2 is constructed to explore distinct spatio-temporal traffic patterns. And the model is analyzed with Grad-CAM to see which parts of the images contribute to the identification of certain traffic patterns, e.g. the network congestion level. Speed predictions are a set of images, of size $(2250, 50, 50)$. Each image is assigned a label, which can be used to train the classifier. The optimizer is Adam with the learning rate and decay rate set to 0.001; the number of epochs is 10 and the batch size is 32. The loss function used is categorical cross entropy for classification problems, and "accuracy" is the metric for measuring the prediction accuracy.

The computer for the experiment is the author's own laptop with an Intel Core i7-10875H @ 2.30GHz processor and an NVIDIA GeForce RTX 2060 GPU. The code is written in Python with TensorFlow Keras on Jupyter Lab for inline visualization.

5

Results

This chapter presents all intermediate and final results produced in the project, as well as corresponding analysis and discussion. Section 5.1 gives a description of the case study network AMSnet, as well as some analyses of the data and corresponding traffic characteristics. The result of data rasterization is also included. In Section 5.2, the findings obtained after feature extraction are discussed. The example results of speed prediction using 3 models and corresponding findings are presented in Section 5.3. In the same section, the comparison of model performances can also be found. Error metrics are analyzed in Section 5.4, including the inspection of error histograms, K-means clustering of error metrics, and the trends of error value with the change of network congestion level. The clustering results for traffic pattern identification are presented, compared, and discussed in Section 5.5. Explanation using Grad-CAM and the insights can be found in Section 5.6.

5.1. Data Rasterization

The AMSnet in latitude-longitude coordinate projection is plotted as shown in Figure 5.1. In total there are 9 roads in this freeway network, namely A10 (north-east), A10 (south-east), A10 (south), A10 (west), A1 (west) & A9 (east), A2, A9, A4, and A9 (west) & A5 (north). The 9 roads are indicated by different colors. Road colors and their lengths are listed in Table 5.1. Figure 5.2 is the x-y coordinate projected network with points partitioning the 9 roads into 201 links, each 400 meters long. The driving directions are also marked in Figure 5.2.

Being partitioned into 400 m links, however, doesn't mean all the links are 400 meters long. The first reason is that some roads are disconnected, and the second reason is that the lengths of roads are not integer multiples of 400. Hence, for each road, there's a link shorter than 400 m. The number of links on each road can be found in Table 5.1 as well. It can be seen from the figures that the roads are not fully consecutive, which might be because the complicated weaving segments (e.g. between A9 and A4) are difficult to completely present.

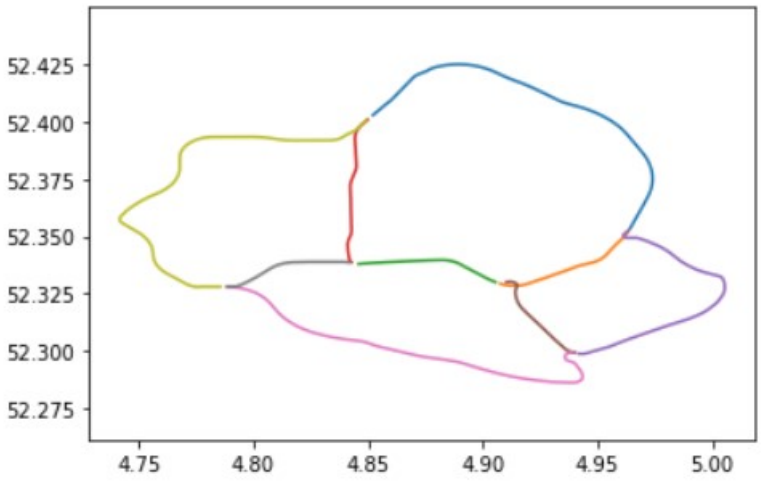


Figure 5.1: AMSnet in lat-long coordinate

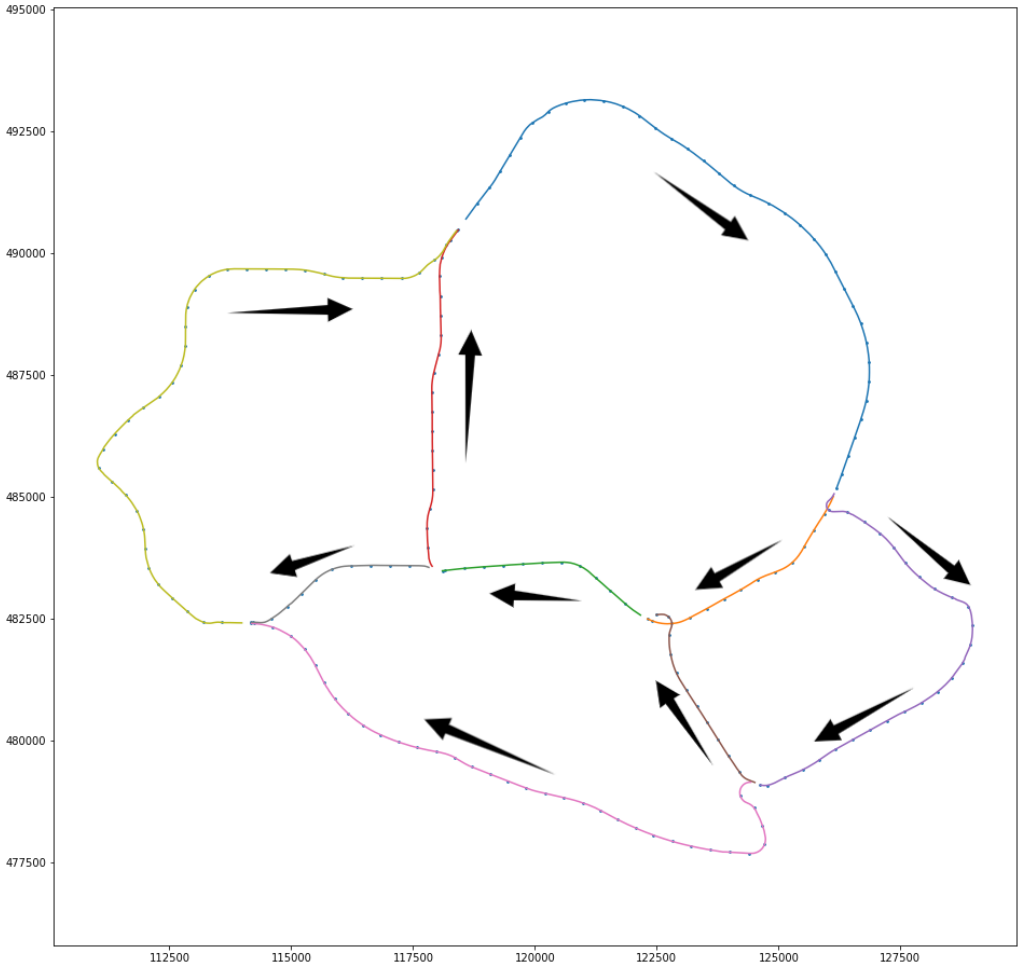


Figure 5.2: AMSnet in x-y coordinate

Table 5.1: Description of studied freeways

Freeway	Index	Color	Length (m)	# links	Cumulative # links
A10 (north-east)	1	Blue	14709.44	37	37
A10 (south-east)	2	Orange	4902.42	13	50
A10 (south)	3	Green	4443.17	12	62
A10 (west)	4	Red	7053.99	18	80
A1 (west) & A9 (east)	5	Purple	10136.57	26	106
A2 (north)	6	Brown	4265.98	11	117
A9 (south)	7	Pink	13648.96	35	152
A4 (east)	8	Grey	4040.70	11	163
A9 (west) & A5 (north)	9	Yellow	15158.33	38	201
Total Length (m)			78359.54		
Total # links			201		

From the raw speed data during the afternoon and the evening peak, a series of speed contour plots can be generated, with one plot indicating the speed pattern for one day. The contour plot of day 78 is shown in Figure 5.3, with the x-axis indicating the time steps and the y-axis indicating the index of links. In Figure 5.3 we can clearly see some congestion propagation patterns during time step 70-140.

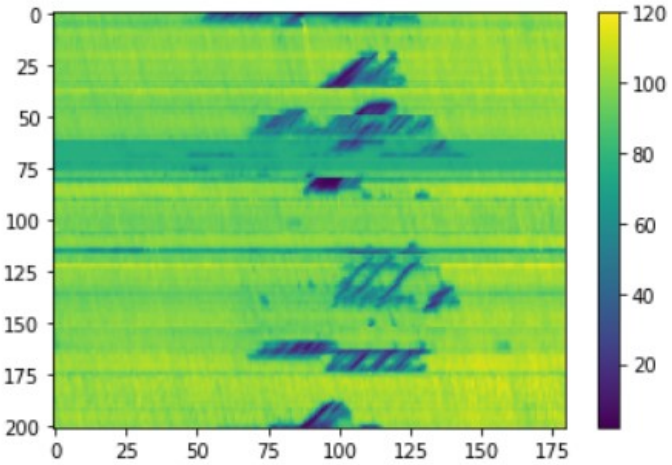


Figure 5.3: Speed contour during afternoon and evening peak on day 78

Take the bottleneck that appears at link 37 between time step 90-120 as an example, the congestion generates and propagates upstream to link 20 for about 6.8 km. According to the cumulative number of links listed in Table 5.1, the congestion occurs on the northeast part of A10 (link 1-37), but its downstream segment on A1 west (link 80-106) is not congested. This bottleneck might be caused by the speed limit on the weaving segment and off-ramp. Besides, by observing the speed values on A10 south (link 50-62), we can conclude that due to the low speed limit on A10 west (link 62-80) and the moving jam between time step 65-90 on A4 east (link 152-163), a moving jam (stop-and-go wave) on A10 south (link 50-62) occurs during time step 95-130.

The speed data is mapped to the network by assigning speed values to points on the roads. Figure 5.4 shows an example of mapped data on AMSnet at time step 120 on day 78. Compared to Figure 5.3, it's obvious that the spatial information is well preserved and easy to understand,

despite it can only show the network traffic state 1 time step per image. Through a further step of processing, the rasterization of speed data is achieved. An example is in Figure 5.5.

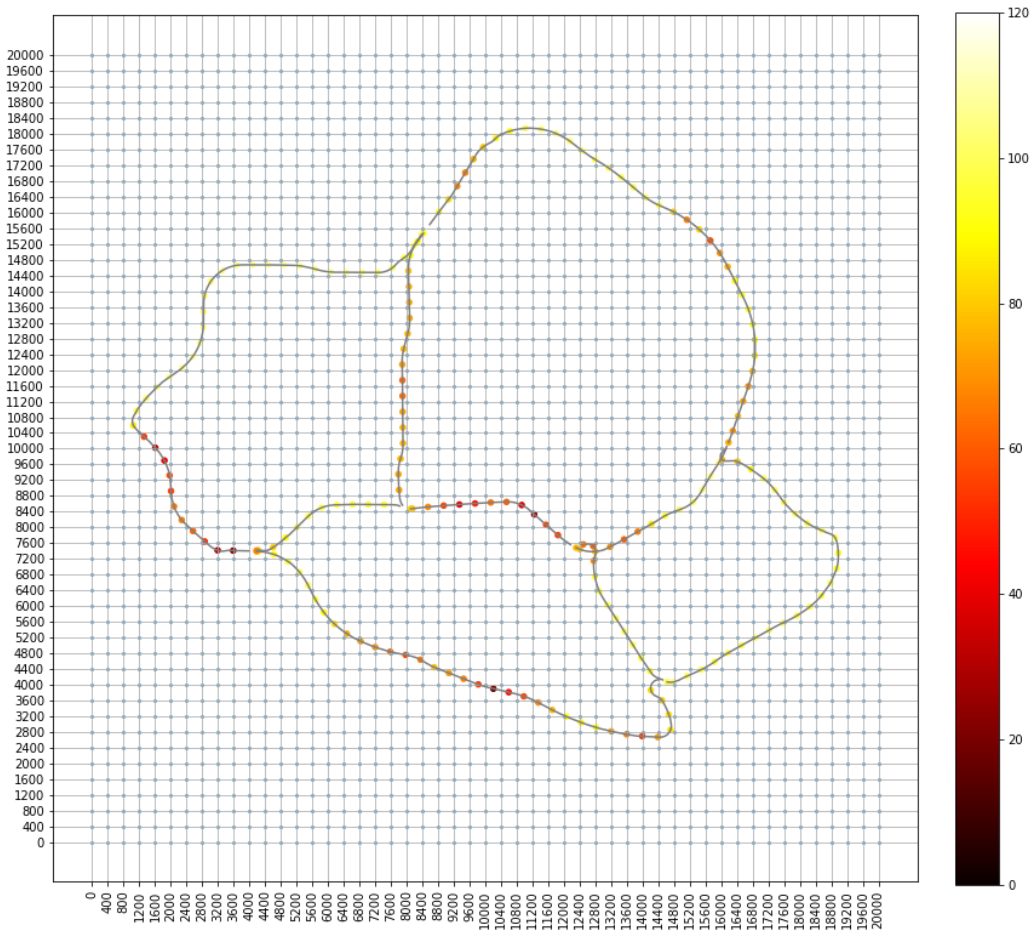


Figure 5.4: Mesh gridded AMSnet at time step 120 on day 78

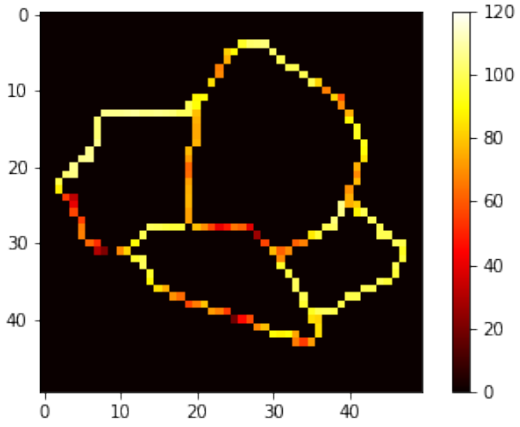


Figure 5.5: Rasterized speed image at time step 120 on day 78

5.2. Feature Extraction

The rasterized images are then fed to Inception ResNet v2 for feature extraction. The last FC layer of the CNN image classifier was removed to have the feature output of size $(18000, 1536)$. A 10-time-step slice of visualized feature vectors is shown in Figure 5.6.

Figure 5.6: Visualized feature vector of 10 time steps

No information can be directly read from the feature vectors, even after the simple visualization. This is because feature vectors are abstractions obtained from the original data, after complex convolution operations the feature output is compressed. Here the author obtains an opinion that simple plotting is not a proper way to visualize the mechanism of convolutional neural networks, which strengthens the necessity for a model explaining method.

So far the positive effect of feature extraction on the model performance has not been exhibited, as the prediction is not performed yet. However, the massive feature extracting time, i.e., 2743 seconds recorded when running the code, is already a quite large number reflecting the possible low time efficiency. Nonetheless, this could be contributed to the hardware limitation, while still being a representative example in the case of implementing the methodology with similar less advanced machines.

5.3. Speed Prediction

Before passing the extracted features to the LSTM encoder-decoder model, it's necessary to convert all the inputs to image sequences. As the task is to predict future 10-time-step speed images with 15-time-step history data, a sliding window method is implemented to generate input sequences from 18,000 images. Figure 5.7 gives an illustration of the mechanism of sliding window, and in our case, 3 sliding steps are tested (25, 10, and 1).

25-step sliding provides each combination of encoder input and decoder output (highlighted in the figure) is completely isolated without overlapping with other sequences; 10-step sliding can produce time-continuous predictions by making decoder output sequences adjacent and not

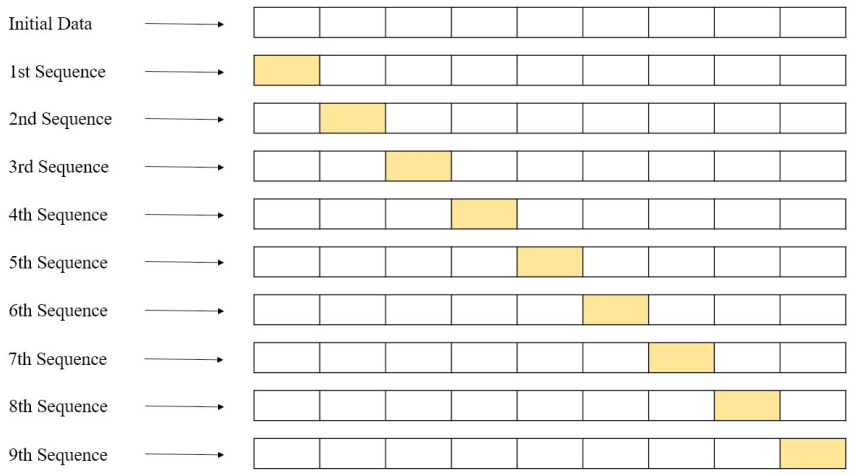


Figure 5.7: An illustration of 25-step sliding window, with 25-step sequences

overlapped; 1-step sliding makes full use of the speed images. But 25-step and 10-step windows cannot fully involve the temporal patterns due to the gap of sampling.

Thus, the plan was determined to use 1-step sliding. However, that would lead to a too-large dataset and too-long model training time which the author's computer cannot handle. In consideration of the balance between computational efficiency and sufficient data amount, finally, a 5-step sliding window was adopted to generate the image sequences used for speed prediction.

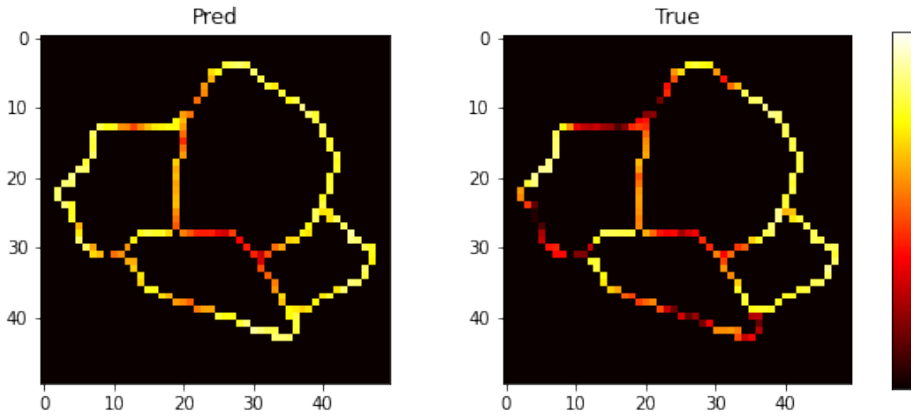


Figure 5.8: Prediction of hybrid model (left) and ground truth (right)

An example of the prediction result given by the hybrid CNN-RNN model and its corresponding ground truth at time step 96 on day 13 is presented in Figure 5.8. Here, the speed values shown in the figures are still standardized, which will be reversed when calculating the error metrics. The occurrences of congestion are partially successfully predicted, for example, the low speed limit on A10 west and the propagated congestion on A10 south. Despite the speed values, this example prediction also fails to accurately indicate the positions of congestion, meaning that the model did not manage to learn from history.

An example of the prediction result given by the LSTM encoder-decoder model and the corresponding ground truth is presented in Figure 5.9. The time step of selected images is the same as Figure 5.8. What can be observed from the comparison between predictions and the ground truth is that the LSTM encoder-decoder performs better in detecting the traffic pattern in images than the hybrid model. The author infers that the reason could be feature extraction removes some useful information. However, the underperformance of the hybrid model may also be due to the lack of tuning.

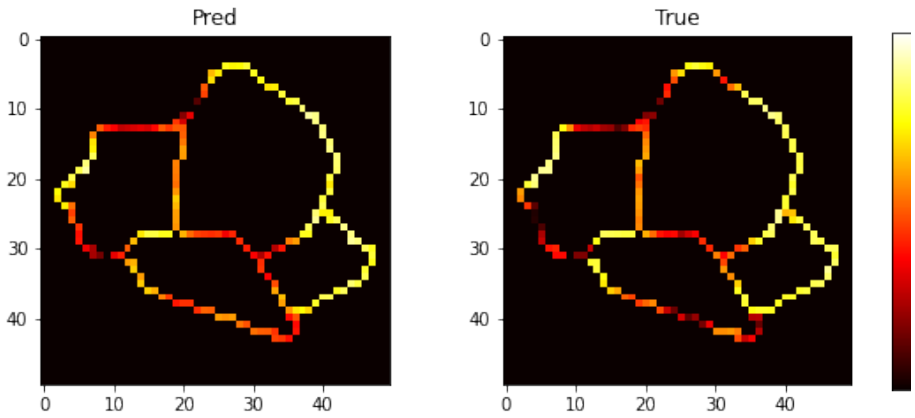
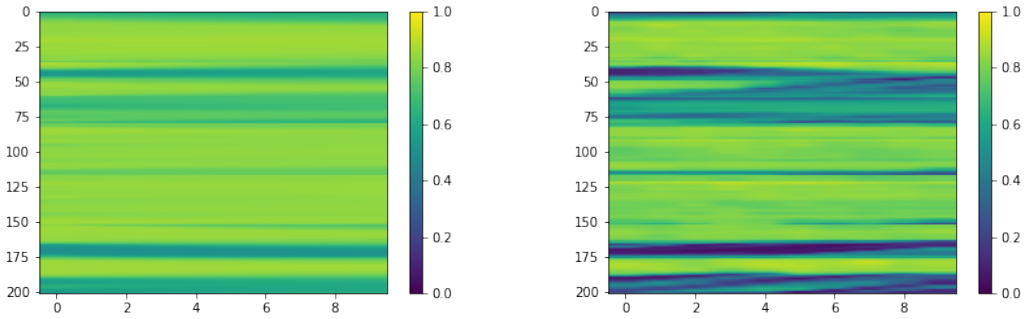


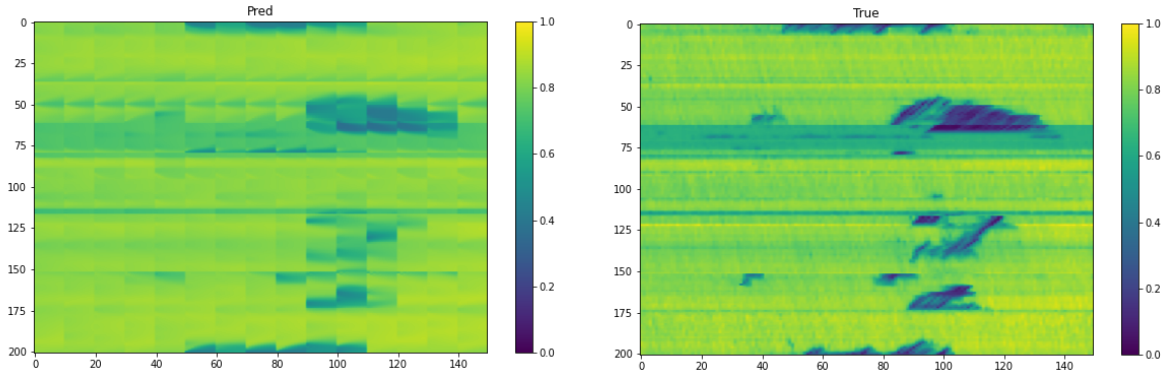
Figure 5.9: Prediction of LSTM encoder-decoder (left) and ground truth (right)

Result example for DGCNN can be found in Figure 5.10. Figure 5.10a appears to be a blurred random spectrum with no apparent meaning. This is because it only presents a prediction of 10 time steps. Below, Figure 5.11 provides the concatenated prediction results for a whole day's afternoon and evening peak, and the corresponding ground truth. Although the predicted values are not very close to the ground truth, it's still easy to observe similar traffic patterns co-occurring at the same locations over such a relatively long time span.



(a) (b)

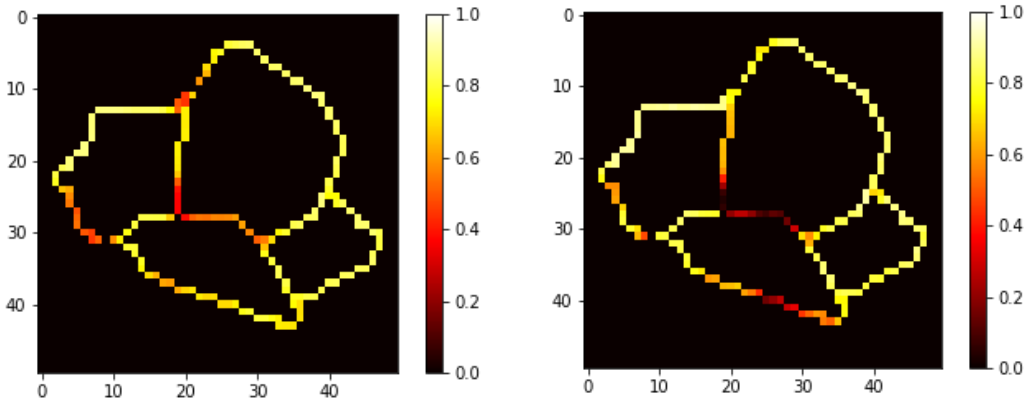
Figure 5.10: Prediction of DGCN (left) and ground truth (right) for 10 time steps



(a) (b)

Figure 5.11: Prediction of DGCN (left) and ground truth (right) for 150 time steps

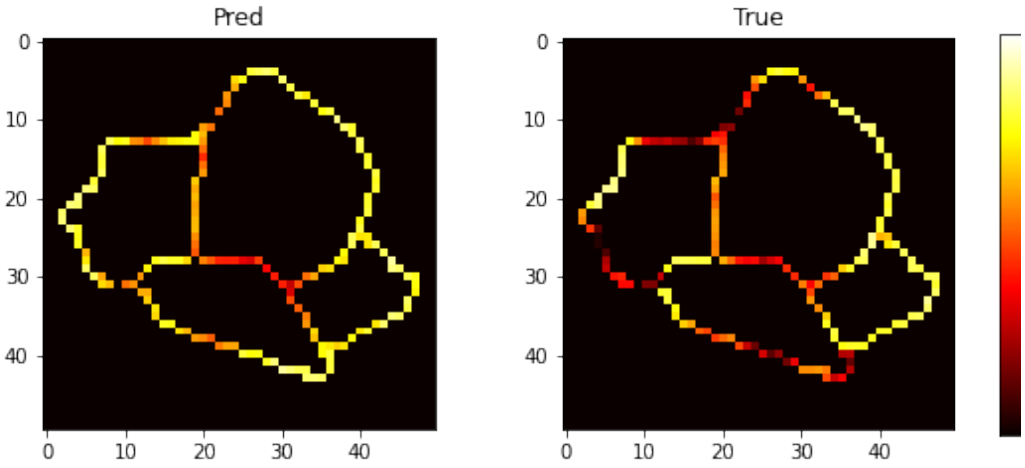
The prediction of DGCN is also transformed to rasterized speed images for a more reasonable comparison with the results of the other 2 models, see Figure 5.12. Comparable to the hybrid model, DGCN is able to predict congestion to some extent, but not with remarkable accuracy. Only congestion trends can be mostly inferred. According to the images, the low speed limit on A10 west can be the major cause of the congestion upstream, as evidenced by the stop-and-go waves on A10 south. This can be further demonstrated by referring to the speed contour between link 50-62 at time step 100 in Figure 5.11b.



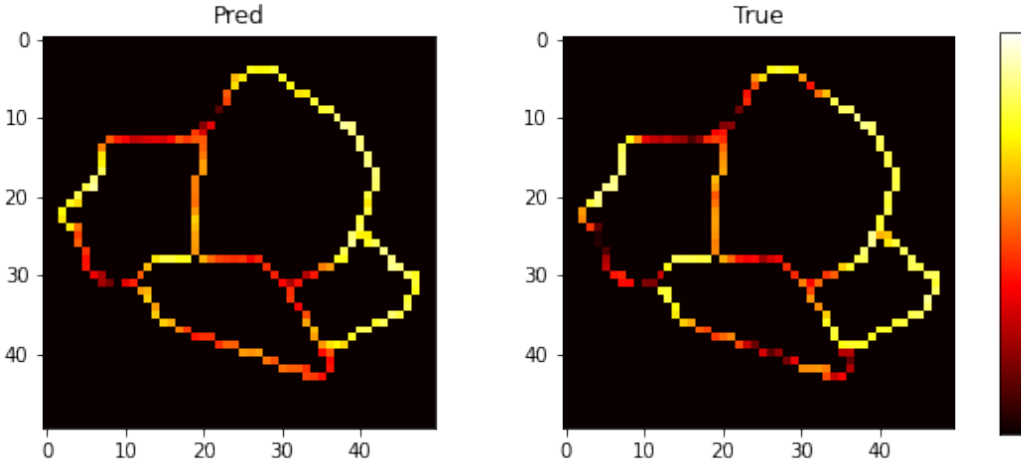
(a) (b)

Figure 5.12: Rasterized prediction of DGCN (left) and ground truth (right) at time step 100

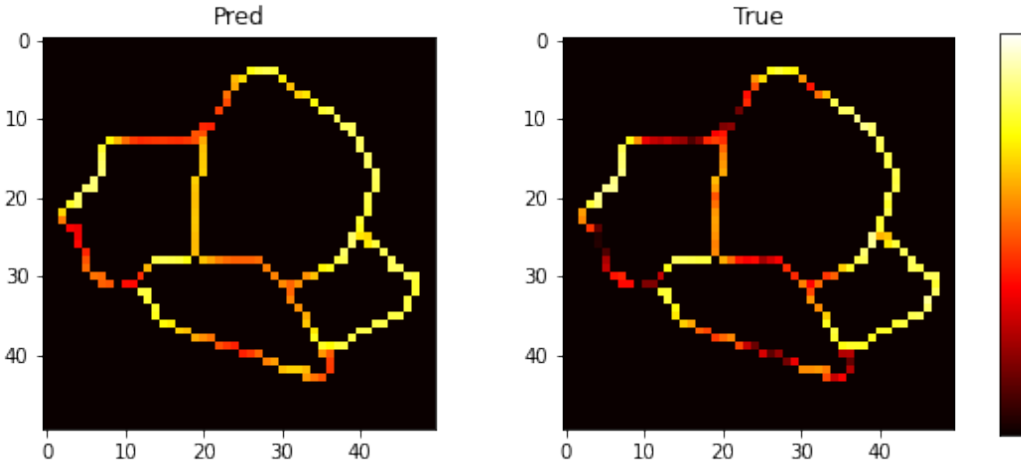
For comparison of all 3 models, the predictions at the same time step 96 on day 13 are shown below in Figure 5.13. The comparison indicates that DGCN prediction best captures the congestion locations among the 3. LSTM encoder-decoder, however, gives a prediction aiming for the best matching values.



(a) Prediction of hybrid model (left) and ground truth (right)



(b) Prediction of LSTM encoder-decoder (left) and ground truth (right)



(c) Prediction of DGCN (left) and ground truth (right)

Figure 5.13: Comparison of 3 models' predictions at time step 96 on day 13

The overall 10-step prediction performance of 3 models using sequences generated with 25-step sliding are listed in Table 5.2. And the prediction error metrics for each of the 10 time steps of the 3 models are respectively presented in Tables 5.3, 5.4, and 5.5. Hybrid CNN-RNN has the lowest MAE and shorter training time than LSTM encoder-decoder, which can be due to its deeper structure and extra feature extraction. However, if the time consumed on feature extraction is considered, the hybrid model is the most computationally expensive one. DGCN has the shortest training time, which means it has extraordinary time efficiency; however, the prediction accuracy is not that satisfying due to too few epochs and the lack of model fine-tuning. Regarding RMSE, the LSTM encoder-decoder gives predictions with the least extent of variation from ground truth, while the RMSEs given by the hybrid model and the DGCN are considerably high for a speed limit of 120 km/h. This means the hybrid model and the DGCN have predictions with larger errors than the LSTM encoder-decoder.

Table 5.2: Overall 10-step prediction performances

Model	MAE (km/h)	RMSE (km/h)	Time (s)
Hybrid CNN-RNN	7.00	12.753	2685 + 2743
LSTM encoder-decoder	5.731	10.811	3476
DGCN	7.396	15.852	490

Table 5.3: Hybrid model prediction performances of each step

Time Steps	0	1	2	3	4	5	6	7	8	9
MAE (km/h)	9.86	6.67	6.63	6.66	6.67	6.68	6.69	6.70	6.73	6.74
RMSE (km/h)	15.16	12.43	12.42	12.46	12.46	12.47	12.45	12.44	12.49	12.49

Table 5.4: LSTM encoder-decoder prediction performances of each step

Time Steps	0	1	2	3	4	5	6	7	8	9
MAE (km/h)	9.43	5.15	5.06	5.19	5.27	5.34	5.38	5.43	5.49	5.55
RMSE (km/h)	16.07	9.29	9.48	9.74	9.92	10.13	10.24	10.39	10.55	10.69

Table 5.5: DGCN prediction performances of each step

Time Steps	0	1	2	3	4	5	6	7	8	9
MAE (km/h)	6.84	6.94	7.07	7.20	7.35	7.50	7.61	7.71	7.80	7.93
RMSE (km/h)	15.36	15.45	15.56	15.69	15.80	15.95	16.03	16.11	16.21	16.32

Figure 5.14 shows the change of error values with the increase of prediction time steps for the 3 models. Both the hybrid model and LSTM encoder-decoder show decreasing trends of MAE value and RMSE value, while DGCN has slightly incremented MAE and RMSE. This might indicate that the hybrid model and LSTM encoder-decoder are more suitable and steadier in terms of longer-period prediction, and DGCN performs better in single-step prediction.

However, this result could also be because the encoder-decoder-based models are not able to capture the underlying patterns in the data, and are making consistent errors throughout the prediction horizon. The uncertainty in the prediction increases with time as well, as previous predictions were used to predict according to the sliding window method, which can lead to

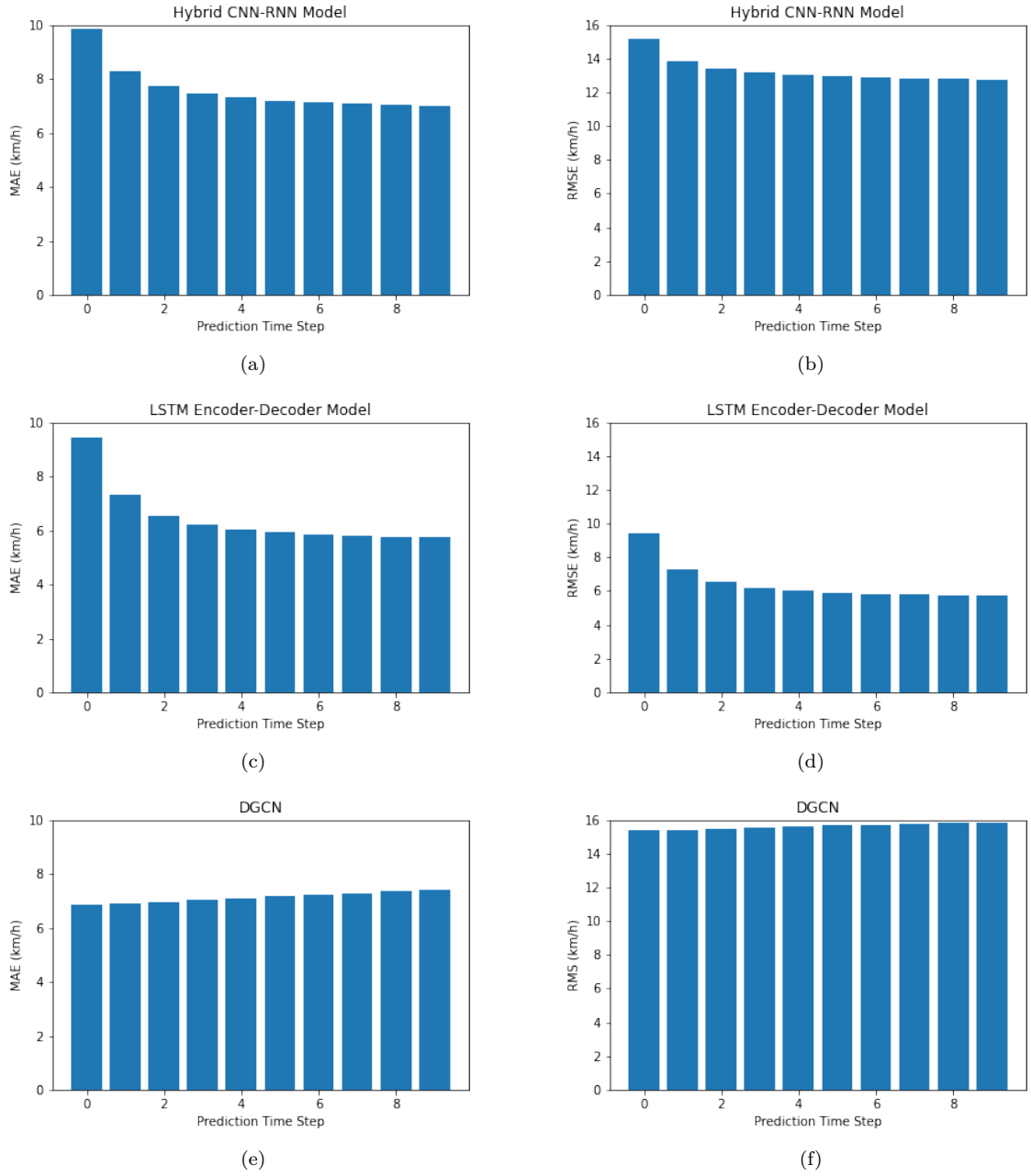


Figure 5.14: MAE and RMSE changes following the increase of prediction time steps

DGCN's rising RMSE. This is because small errors in the prediction at earlier time steps can compound and magnify as the prediction horizon gets longer and more predictions are used to predict.

Observing the evolution in error metrics at every single step following the increase in time (listed in Table 5.3 and 5.4), for both the hybrid model and the LSTM encoder-decoder, the errors first decrease and then increase. This finding is notable since it is commonly expected that RMSEs and MAEs will increase with time, as observed in the results of DGCN (Table 5.5), due to previously predicted results being used as later predictors. One possible explanation for this phenomenon could be the mechanism of the encoder-decoder (Figure 3.5), where the decoder requires a trigger to initiate predictions, and the trigger used by the author was an all-0

matrix. It is possible that using a more appropriate trigger could enhance the performances of encoder-decoder-based models.
As each sequence of prediction is 10 images at 10 consecutive time steps, they need to be concatenated together for a long period of prediction observation. One thing notable is that the selected sliding window is 5 steps, so in each predicted sequence 5 steps of predictions are extracted and attached together for long-period prediction. Otherwise, there will be coincided multiple results for most of the time steps. Taking the 10th prediction sequence on day 9 as an example, the prediction images given by the 3 models and corresponding ground truths can be found in Figure 5.15, 5.16, and 5.17.

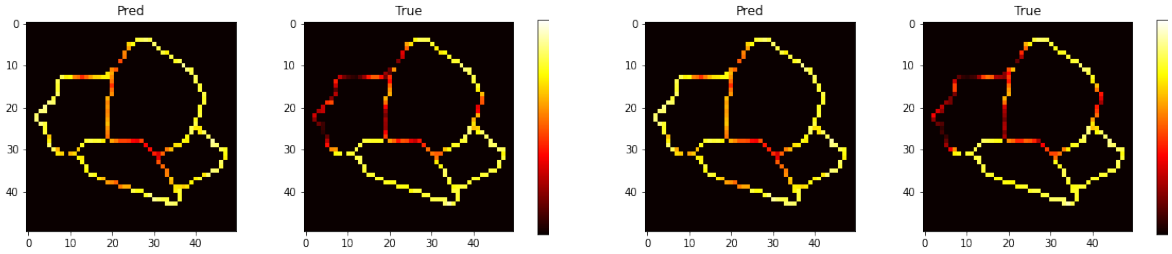
Compared to LSTM encoder-decoder and DGCN, the hybrid CNN-RNN model didn't predict very well, because:

1. Some of the congestion areas were not identified and predicted, while some low/no congestion areas were mistakenly predicted to have low speeds. This directly shows the hybrid model has fairly lower accuracy than the other 2 benchmarks.
2. The predicted congestion didn't propagate upstream along the time, indicating the model failed to capture the spatio-temporal characteristics of the traffic. A typical example is the moving jam on northeast A10 propagating from the east to the north, which the hybrid model failed to predict, and the LSTM encoder-decoder partially learned.
3. The predictions for 10 time steps, i.e. 20 minutes, almost remained the same. If the congestion was caused by unexpected or emergent accidents, then the underperformance might be caused by the changes in the underlying patterns that the model needs to capture; if the congestion was caused by bottlenecks, then it shows that the model is not capable of finding the patterns in historical data.

However, no matter what the other reasons are, it is notable that none of these can shake the fact that the hybrid model did not manage to learn the underlying traffic dynamics.

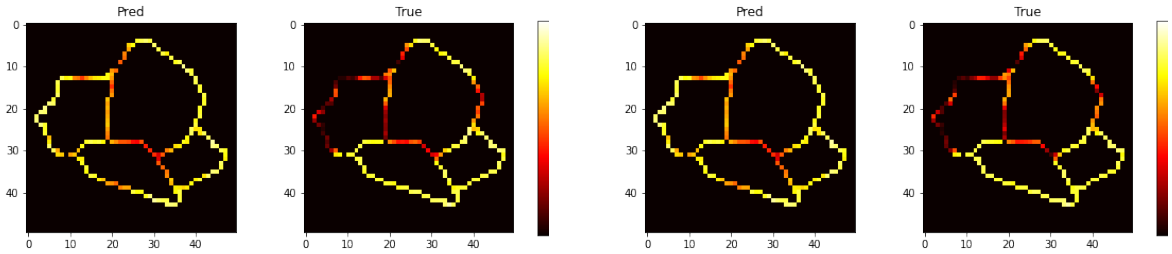
Although LSTM encoder-decoder predictions have the lowest overall error values, it cannot be ignored that there's a sudden shift that appeared on A10 east at time step 94 and 95, and the predicted congestion didn't propagate upstream along the time either. In terms of prediction accuracy, DGCN may not have the most favorable numerical values. However, it demonstrates the ability to provide more realistic predictions by depicting slight congestion propagation and corresponding congestion regions that align with the actual situation.

These findings, despite the fact that the models were not fine-tuned, provide some support to the argument that graph structure is more suitable for traffic network speed prediction than image by efficiently preserving network topology. Besides, pretrained model-based feature extraction indeed reduces the dimensionality and complexity of the data, but the procedure also brings the risk of losing important information.



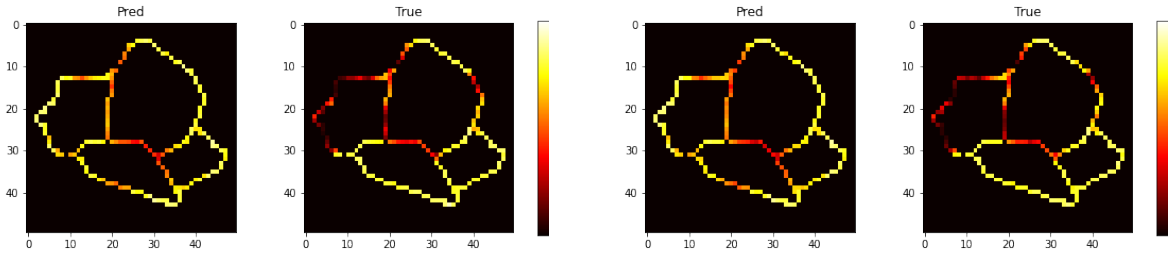
(a) Day 9 time step 90

(b) Day 9 time step 91



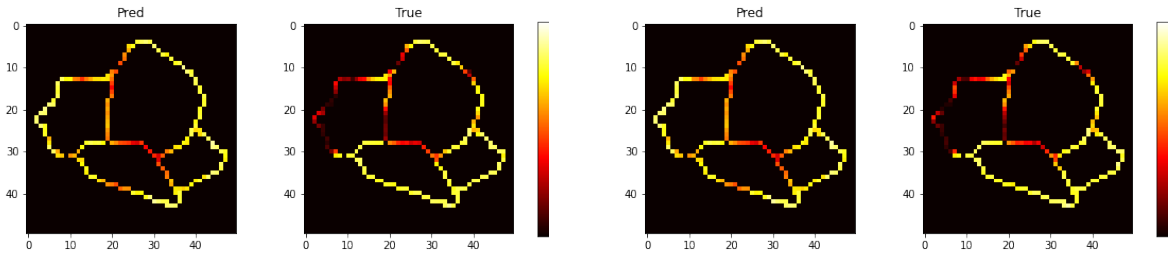
(c) Day 9 time step 92

(d) Day 9 time step 93



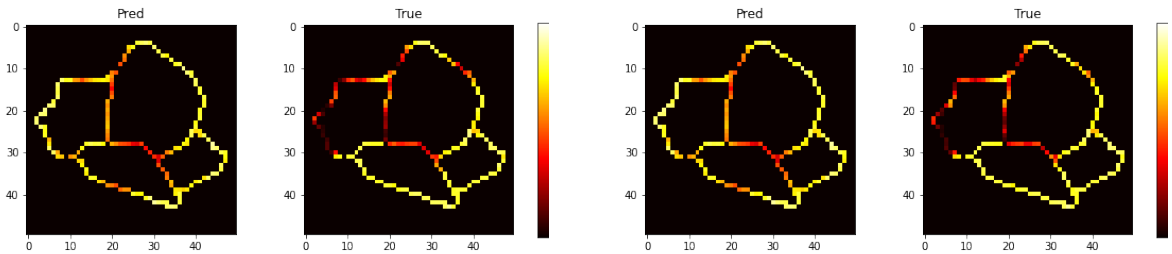
(e) Day 9 time step 94

(f) Day 9 time step 95



(g) Day 9 time step 96

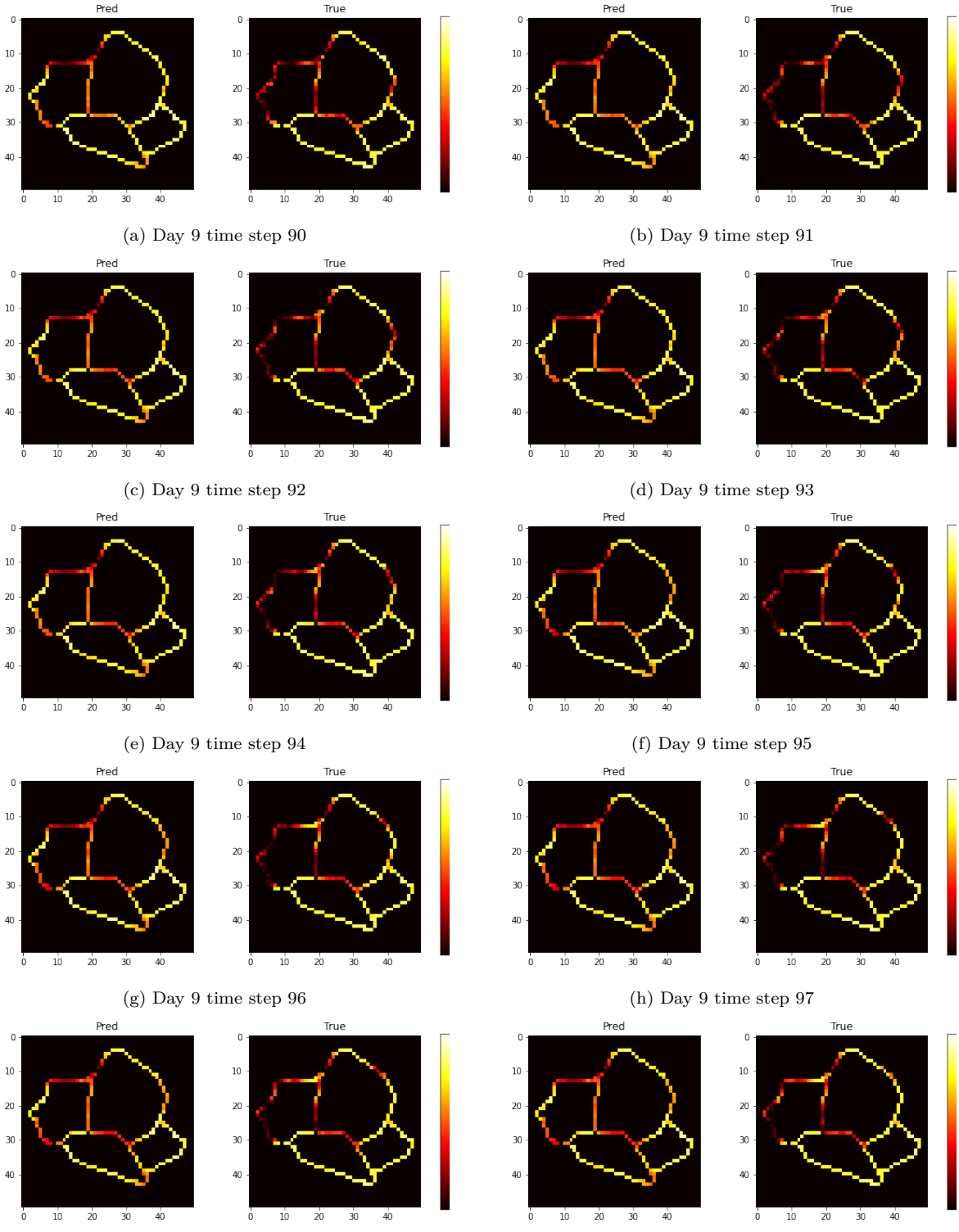
(h) Day 9 time step 97



(i) Day 9 time step 98

(j) Day 9 time step 99

Figure 5.15: Predictions of hybrid CNN-RNN model (left) and ground truths (right) from time step 90 to 99 on day 9



(a) Day 9 time step 90

(b) Day 9 time step 91

(c) Day 9 time step 92

(d) Day 9 time step 93

(e) Day 9 time step 94

(f) Day 9 time step 95

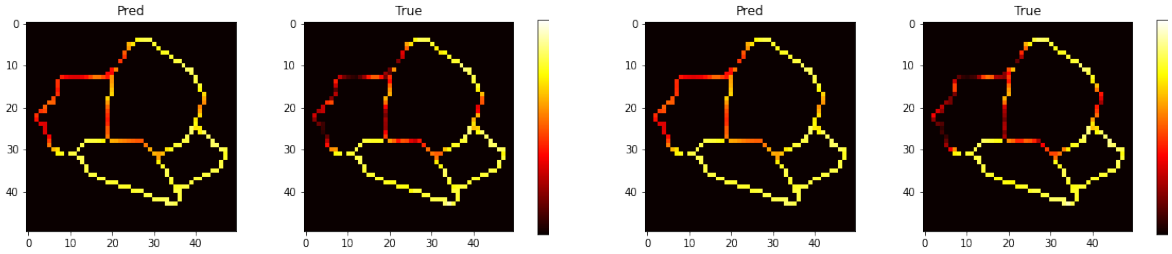
(g) Day 9 time step 96

(h) Day 9 time step 97

(i) Day 9 time step 98

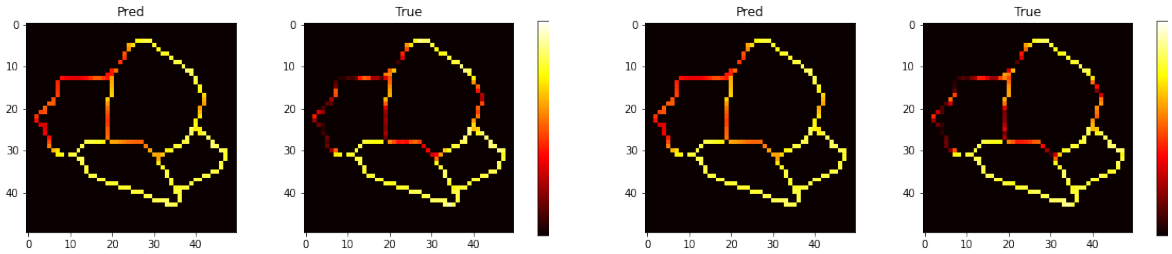
(j) Day 9 time step 99

Figure 5.16: Predictions of LSTM encoder-decoder (left) and ground truths (right) from time step 90 to 99 on day 9



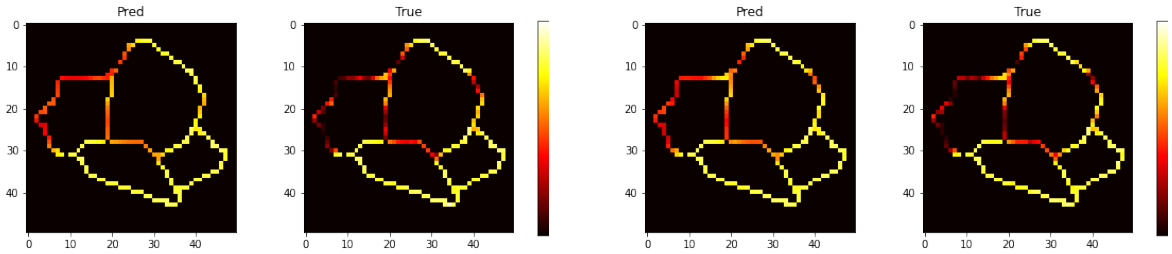
(a) Day 9 time step 90

(b) Day 9 time step 91



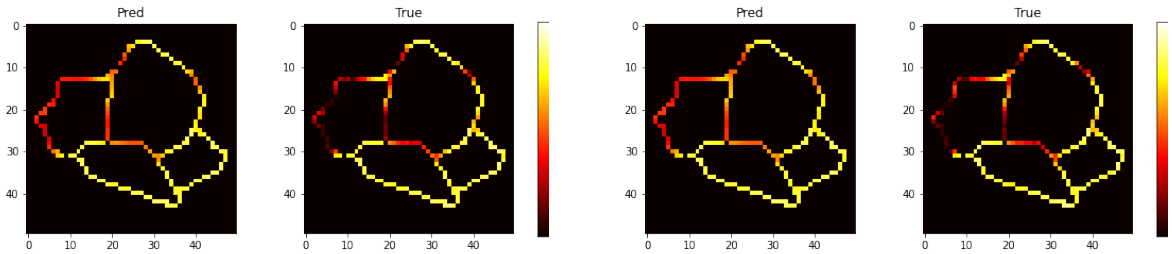
(c) Day 9 time step 92

(d) Day 9 time step 93



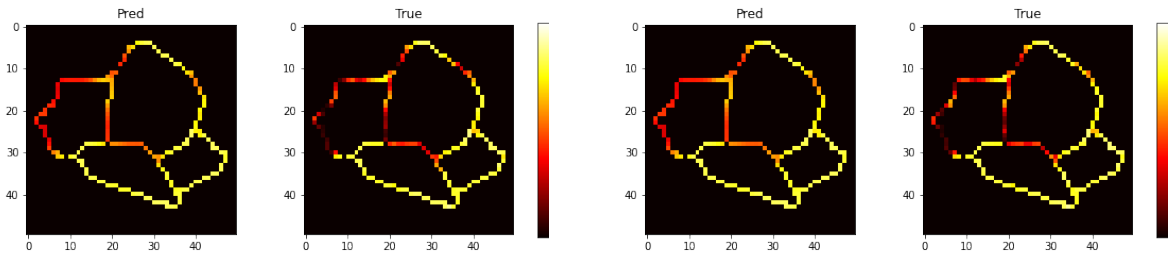
(e) Day 9 time step 94

(f) Day 9 time step 95



(g) Day 9 time step 96

(h) Day 9 time step 97



(i) Day 9 time step 98

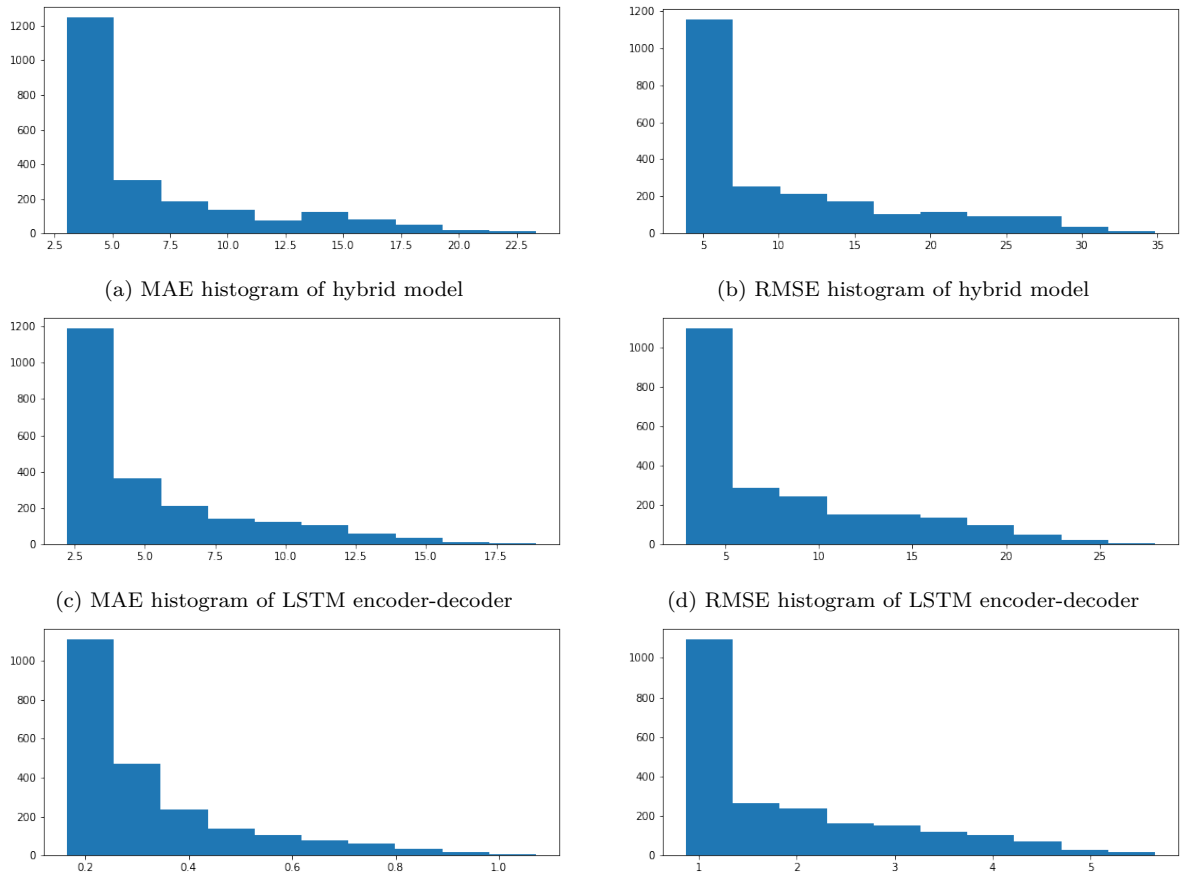
(j) Day 9 time step 99

Figure 5.17: Predictions of DGCN (left) and ground truths (right) from time step 90 to 99 on day 9

5.4. Error Analysis

This section mainly analyzes the image-specific error metrics of speed prediction for the 3 models. The distributions of MAE and RMSE are plotted, and the error values are clustered in order to find underlying patterns. Scatter plots of mean speed and RMSE value are also plotted to observe the relationship between the two variables.

Below (Figure 5.18) presents the MAE and RMSE distribution histograms for the 3 models with bin=10. The MAEs and RMSEs of hybrid model predictions spread the widest, which means the portion of its single predictions have a larger bias from ground truth is larger than other models' single predictions. While the error metrics of DGCN show that its single predictions are the closest to the ground truth values.



(a) MAE histogram of hybrid model

(b) RMSE histogram of hybrid model

(c) MAE histogram of LSTM encoder-decoder

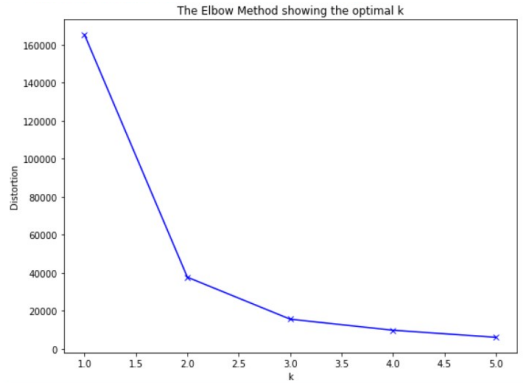
(d) RMSE histogram of LSTM encoder-decoder

(e) MAE histogram of DGCN

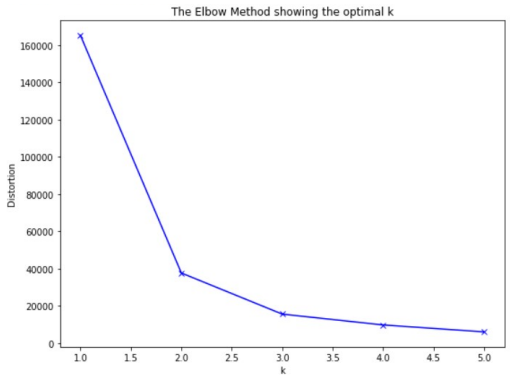
(f) RMSE histogram of DGCN

Figure 5.18: Error metric distributions of the 3 models

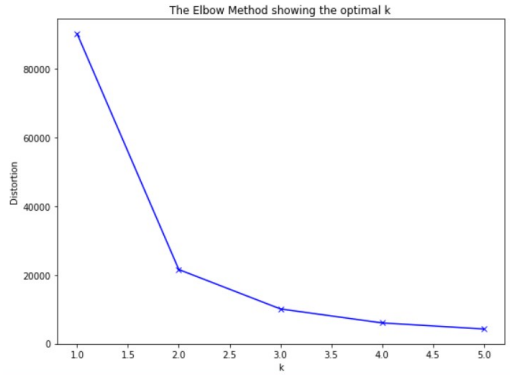
As histogram can only allow for the inspection of one error metric, K-means clustering is applied to combine the 2 error metrics. 2 initialization methods, namely K-means ++ and random, are chosen for clustering and comparison. First, the elbow method is used for finding the optimal K value for clustering, see Figure 5.19.



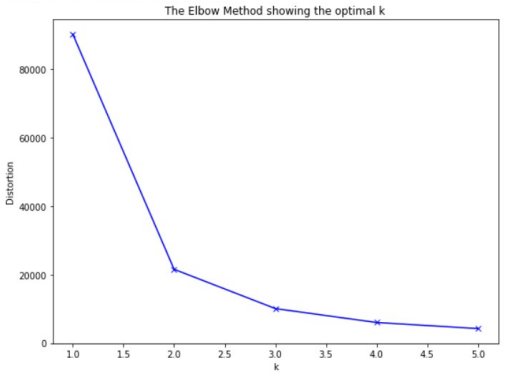
(a) K-means ++, hybrid model



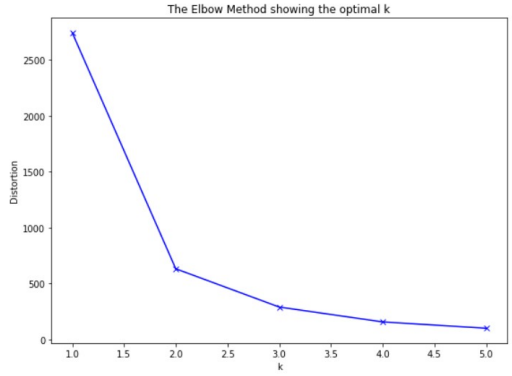
(b) Random, hybrid model



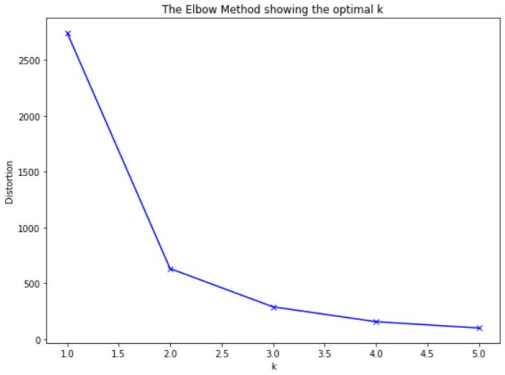
(c) K-means ++, LSTM encoder-decoder



(d) Random, LSTM encoder-decoder



(e) K-means ++, DGCN



(f) Random, DGCN

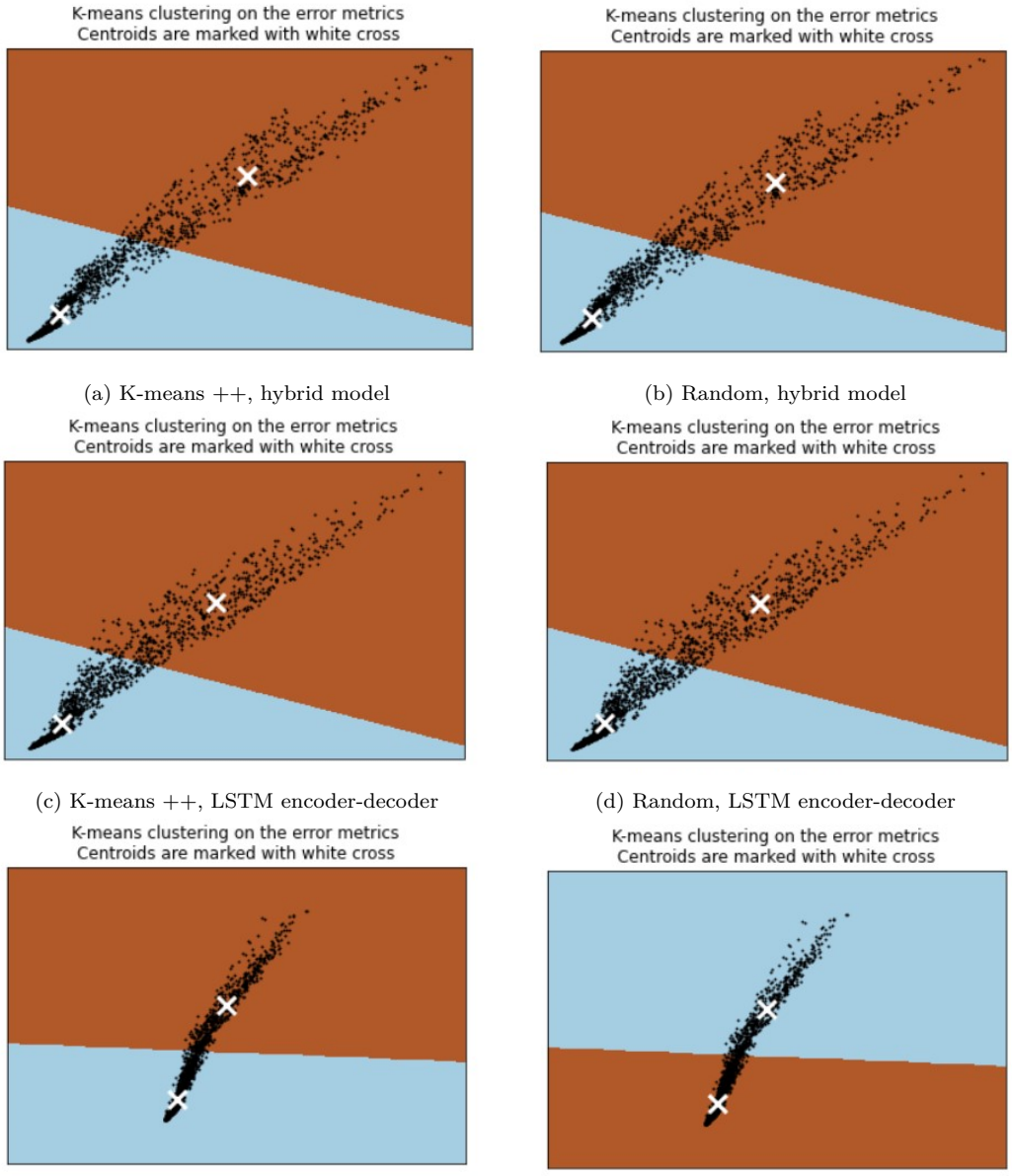
Figure 5.19: Elbow plots of 2 initialization methods for the 3 models

It's clear that the optimal K in all the cases is 2. Therefore, the MAE-RMSE matrices are clustered with K=2. Results are shown below (Figure 5.20 and Table 5.6). And it can be observed from the plots and the table that the 2 different initialization methods didn't bring different results. The clustering itself also just divides the data into 2 parts (smaller and larger), as MAE and RMSE are dependent on each other, thus no insights can be gained. The comparison of prediction and ground truth from each cluster also supports this argument, taking the prediction of the hybrid model as an example (Figure 5.21).

Table 5.6: Cluster distribution of random initialization K-means for the 3 models

Cluster	1	0
Hybrid	1733	517
LSTM	1694	556
DGCN	1752	498

Cluster 1 represents the cluster with lower MAE and lower RMSE, and cluster 0 is the opposite. From Figure 5.21a and 5.21b, it's obvious that the source of large errors is in the southeast part of the network, where the quite severe congestion was not successfully predicted by the hybrid model. While Figure 5.21c and 5.21d are considerably similar to each other, the errors are naturally small.



(a) K-means ++, hybrid model

(b) Random, hybrid model

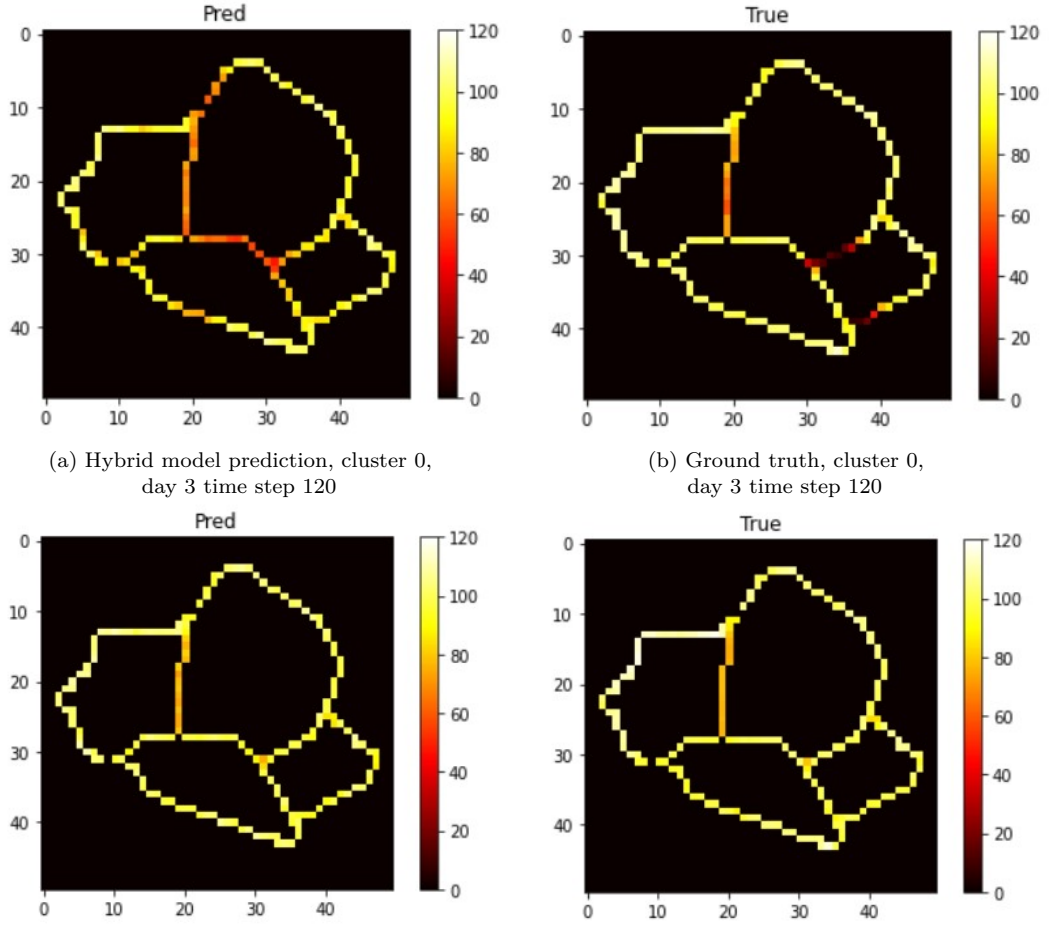
(c) K-means ++, LSTM encoder-decoder

(d) Random, LSTM encoder-decoder

(e) K-means ++, DGCN

(f) Random, DGCN

Figure 5.20: Clustering results of 2 initialization methods for the 3 models



(a) Hybrid model prediction, cluster 0, day 3 time step 120

(b) Ground truth, cluster 0, day 3 time step 120

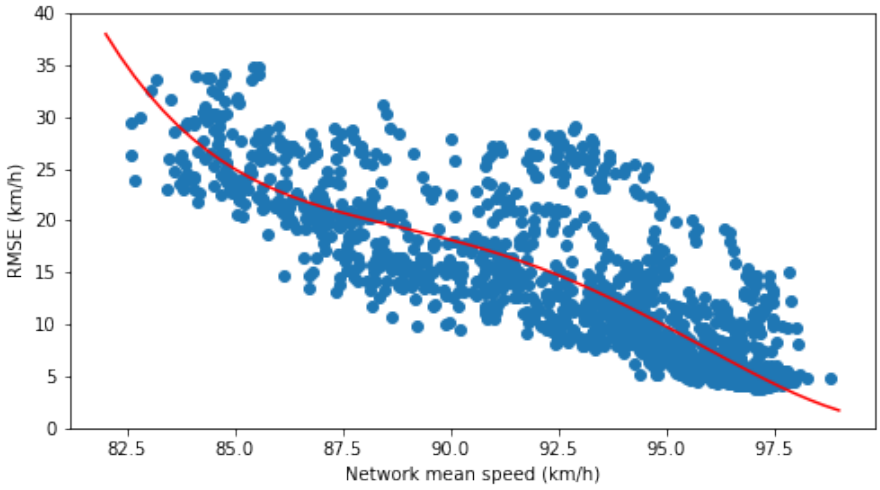
(c) Hybrid model prediction, cluster 1, day 0 time step 126

(d) Ground truth, cluster 1, day 0 time step 126

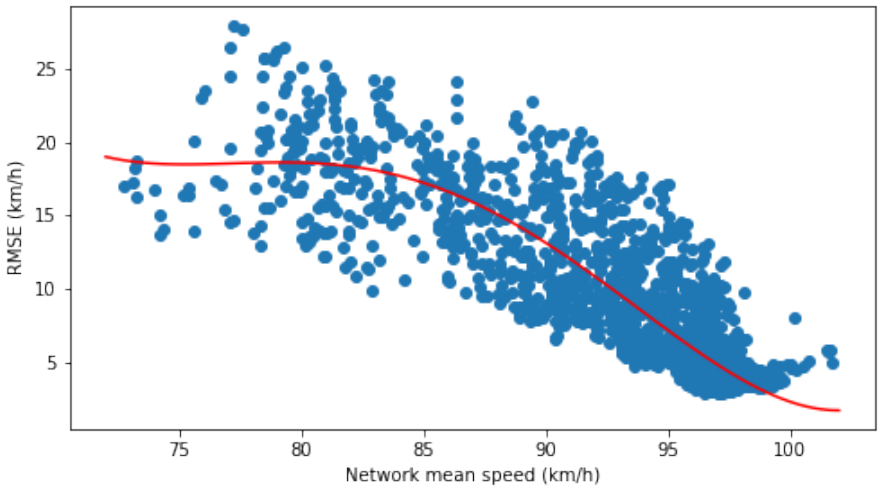
Figure 5.21: Clustering results example & corresponding ground truth

After the exploration of the relationship between MAE and RMSE, the author found they are dependent on each other. Hence, from here on, RMSE only is taken as the error metric for brevity. Except for observing error metrics themselves, the relationship between the network congestion level and error level can also be a point to dig into. Below figures (Figure 5.22) are the scatter plots of network mean speed (x-axis) and RMSE value (y-axis) for each prediction, and a line is fitted to each figure for inspecting the trend.

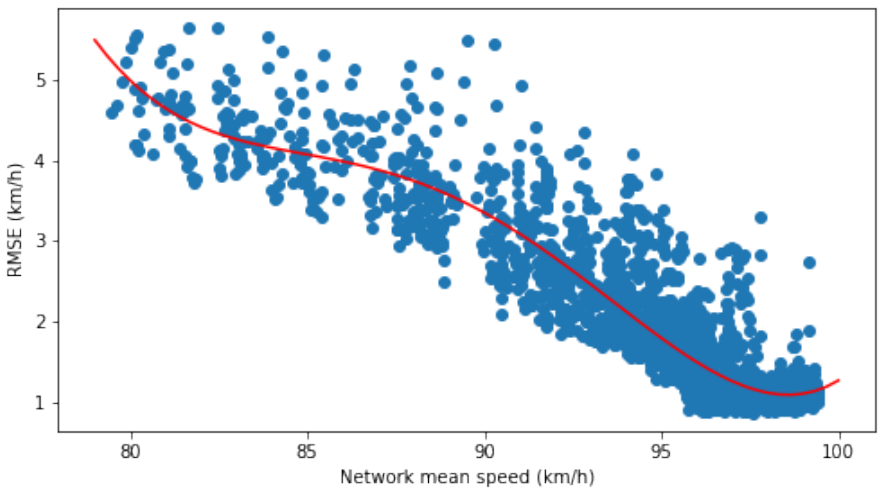
All 3 subfigures show that the higher the network mean speed, the lower the RMSE value. This can indicate the models are better at predicting low/non congestion situations, but not that good at capturing the information during congested times. Another finding is that there's an "elbow" on each fitted line. If the network average speed decreases from 97 km/h to 92 km/h, the hybrid model's RMSE will increase by 10.19 km/h; for a decrease from 90 km/h to 85 km/h, the RMSE increases by 6.79 km/h. Results for LSTM encoder-decoder and DGCN can be found in Table 5.7.



(a) Hybrid model



(b) LSTM encoder-decoder



(c) DGCN

Figure 5.22: Relationship between network congestion level and error level

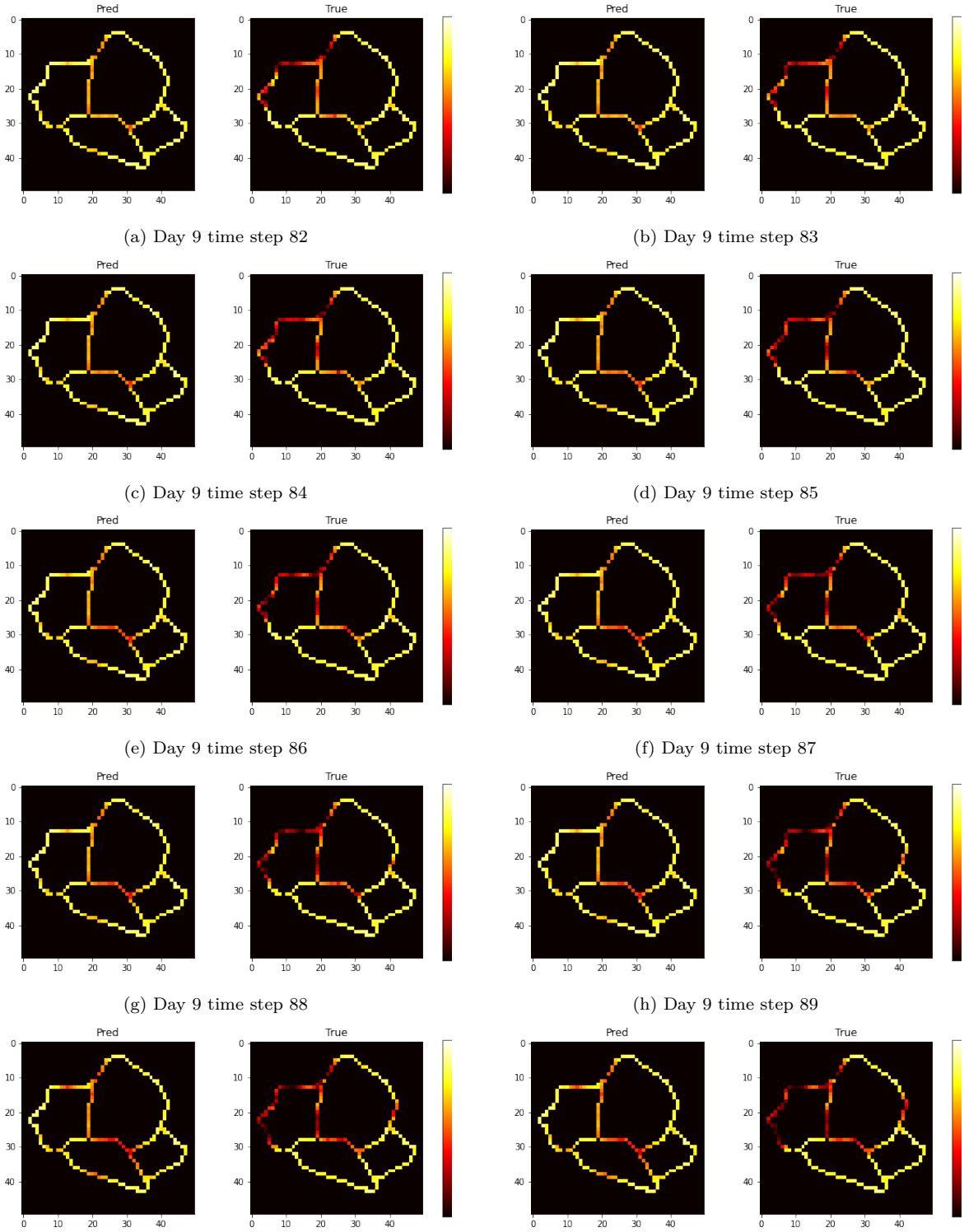
Table 5.7: RMSE increase with speed decrease

Speed Decrease km/h	97-92	90-85
RMSE Increase (Hybrid)	10.19	6.79
RMSE Increase (LSTM)	5.97	4.07
RMSE Increase (DGCN)	1.53	0.72

This conveys a message, i.e., when the network is less congested, the predicting performances of the 3 models are more sensitive to the change of network congestion states; while when the network becomes more congested, e.g. network mean speed is smaller than 87 km/h, the models' performances are relatively steady. The reason could be because the free flow situation is more uniform, but the congested situation contains a large number of instant changes, whose complexity prevents the deep learning models learn from the historical data. Again, the results highlight the good prediction accuracy of DGCN, however, it's also interesting to find that LSTM encoder-decoder generates the most abundant traffic states (network mean speed goes from 72 to 102, which is the largest range among the 3).

Digging deeper into sequences predicted by the models, we can refer to Figure 5.23, 5.24, and 5.25 for the sequence right before the prediction with largest error given by hybrid CNN-RNN model, LSTM encoder-decoder and DGCN. From the figures, it can be easily found that the ability of the hybrid model on predicting congestion is much weaker than the other 2 models, thus also resulting in the higher RMSE values and larger RMSE increase reflected in Figure 5.18b, Figure 5.22a, and Table 5.7.

Referring to Figure 5.23g - 5.23j, a congestion propagation at around (12,12) can be observed. However, there's a large flaw here: the direction of congestion propagation is the same as the driving direction, which is not possible, as the jam always propagates upstream. This turns out a drawback of simply transforming the traffic data to static images, which is the deep neural network may not be able to learn the driving direction from the data.



(a) Day 9 time step 82

(b) Day 9 time step 83

(c) Day 9 time step 84

(d) Day 9 time step 85

(e) Day 9 time step 86

(f) Day 9 time step 87

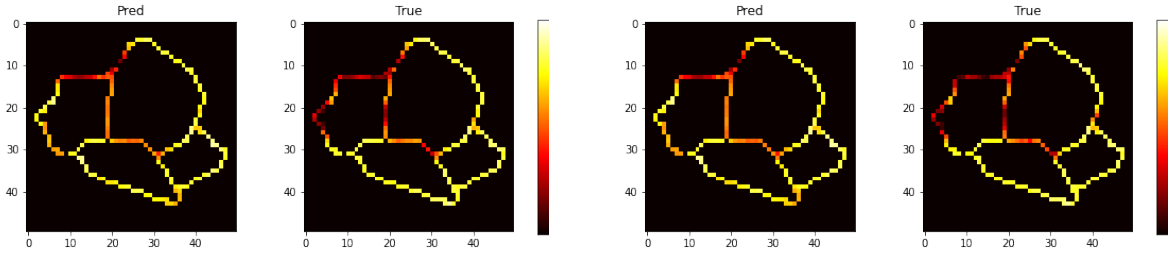
(g) Day 9 time step 88

(h) Day 9 time step 89

(i) Day 9 time step 90

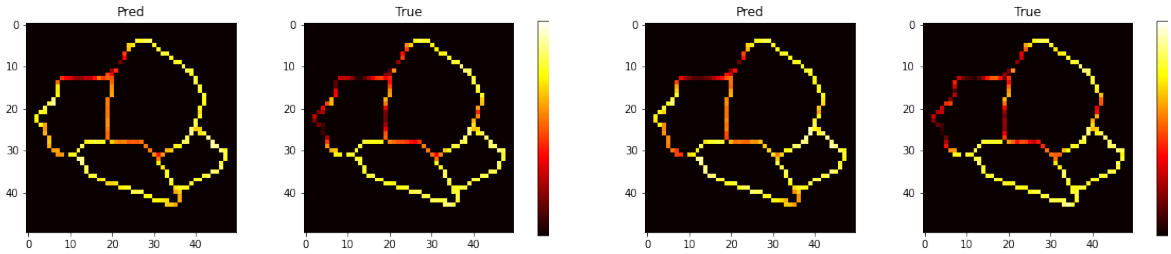
(j) Day 9 time step 91

Figure 5.23: Predictions of hybrid CNN-RNN model (left) and ground truths (right) of sequence before the largest RMSE prediction



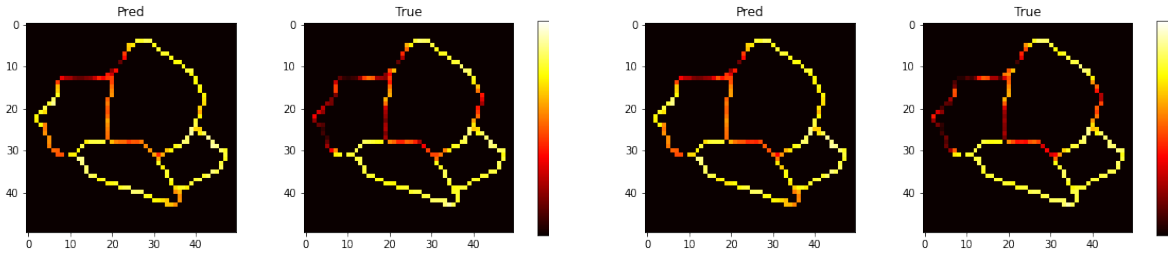
(a) Day 9 time step 87

(b) Day 9 time step 88



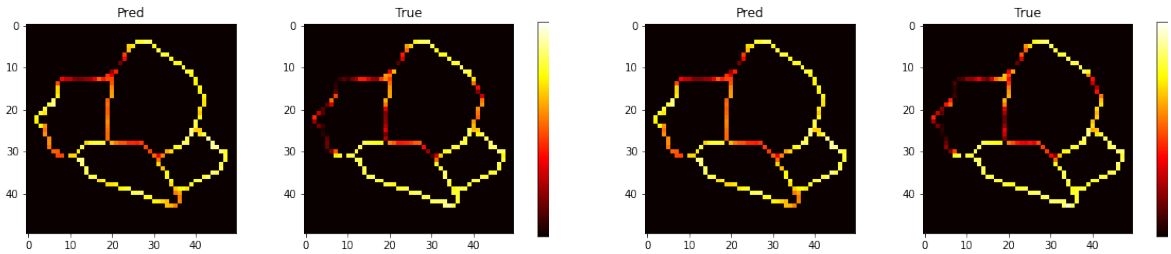
(c) Day 9 time step 89

(d) Day 9 time step 90



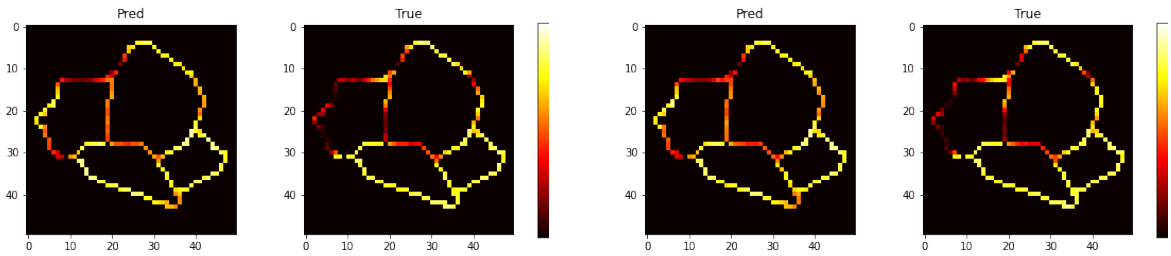
(e) Day 9 time step 91

(f) Day 9 time step 92



(g) Day 9 time step 93

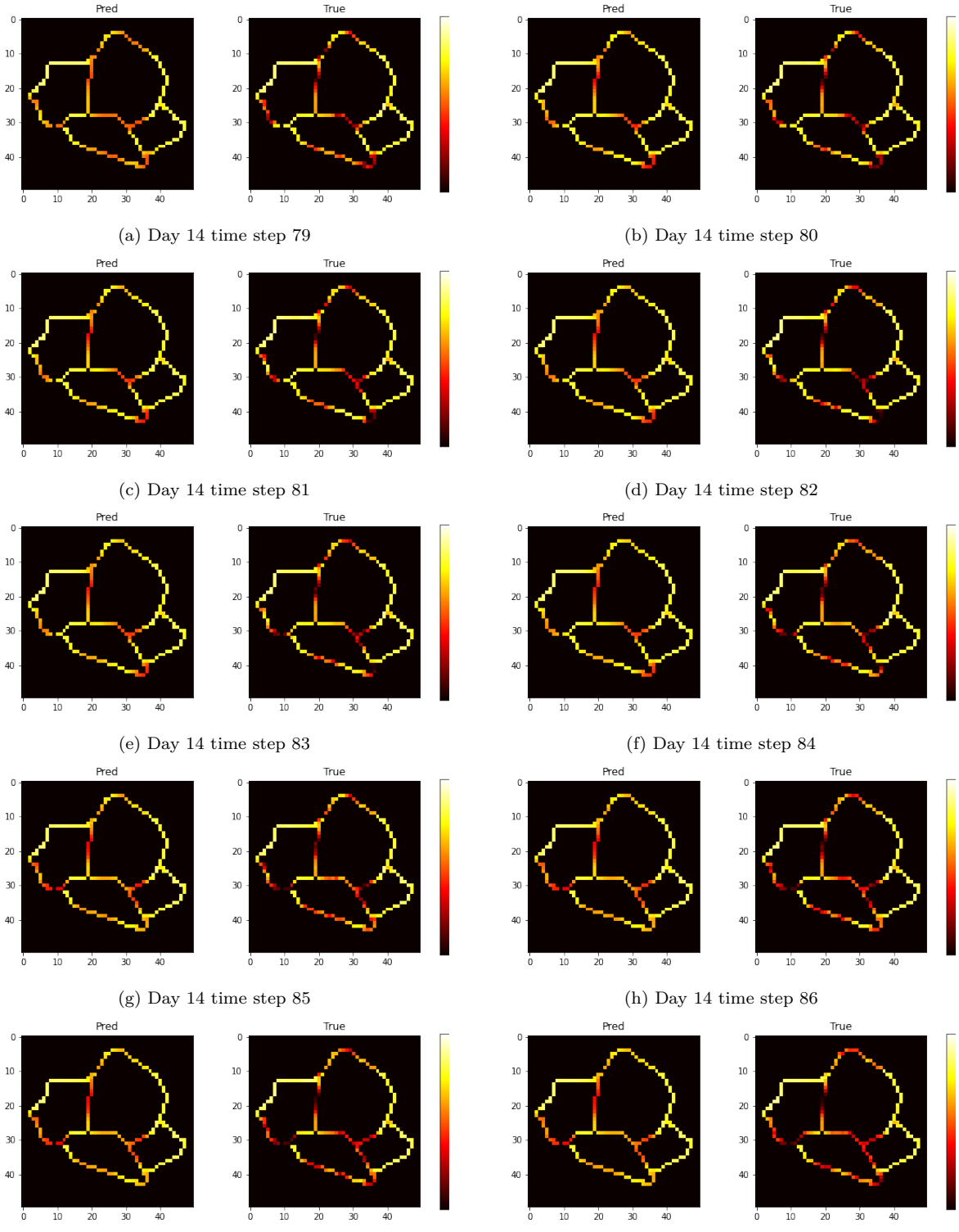
(h) Day 9 time step 94



(i) Day 9 time step 95

(j) Day 9 time step 96

Figure 5.24: Predictions of LSTM encoder-decoder (left) and ground truths (right) of sequence before the largest RMSE prediction



(a) Day 14 time step 79

(b) Day 14 time step 80

(c) Day 14 time step 81

(d) Day 14 time step 82

(e) Day 14 time step 83

(f) Day 14 time step 84

(g) Day 14 time step 85

(h) Day 14 time step 86

(i) Day 14 time step 87

(j) Day 14 time step 88

Figure 5.25: Predictions of DGCN (left) and ground truths (right) of sequence before the largest RMSE prediction

5.5. Traffic Pattern Identification

As mentioned in Chapter 4, each image will be assigned a label for classification. By calculating the network mean speed, the prediction images can be divided into 3 classes: low congestion, mild congestion, and high congestion. The network mean speed values of ground truths lie in the range (72,102), so the first 30% in this range, i.e., 72-81 km/h, is high congestion. 30% to 70% (81-93 km/h) is mild congestion; 70% and larger (93-102 km/h) belongs to low congestion level. The prediction images are then labeled accordingly. The distribution of congestion levels in predictions of the 3 models can be found in Table 5.8 below.

Table 5.8: Distribution of congestion levels

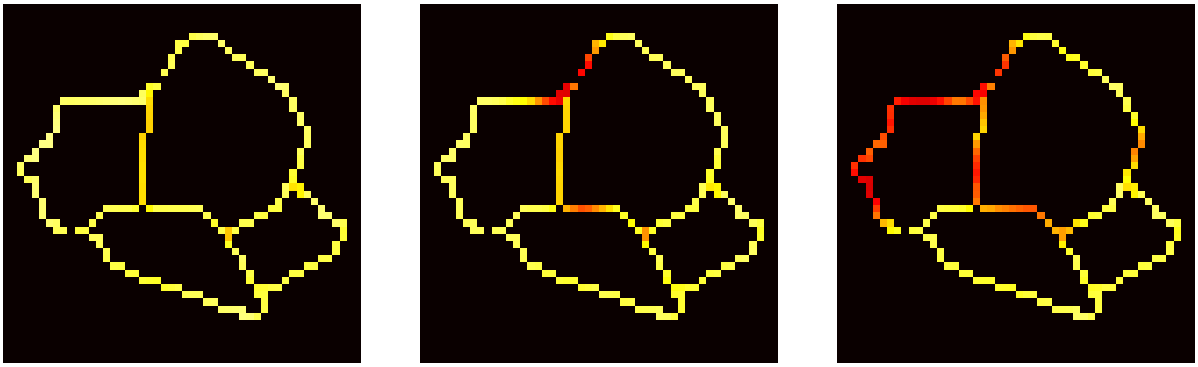
Class (Hybrid)	Low congestion	Mild congestion	High congestion	Total
Number of predictions	1660	590	0	2250
Class (LSTM)	Low congestion	Mild congestion	High congestion	Total
Number of predictions	1598	544	108	2250
Class (DGCN)	Low congestion	Mild congestion	High congestion	Total
Number of predictions	1700	530	20	2250

As each generated prediction image i will be possibly used to predict next sequences, the n error values of "prediction i's prediction sequence I_n" can be attached to I. In our case, $n = 3$. Therefore, each prediction I will have an RMSE distribution, in which the RMSE values are the 3 mean RMSEs coming from sequences I_1, I_2, and I_3, which are predicted using image i. Similarly, the aggregation of RMSE distribution can also be used to classify the images into different error classes: low prediction error, medium prediction error, and high prediction error. Aggregated RMSEs lie in the range (0,34), with 30% and 70% used to divide the classes. Hence, low prediction error means RMSE = 0-10.2 km/h; medium prediction error is 10.2-23.8 km/h; high prediction error is 23.8-34 km/h. The distribution of error classes in predictions of the 3 models can be found in Table 5.9 below.

Table 5.9: Distribution of error classes

Class (Hybrid)	Low error	Mild error	High error	Total
Number of predictions	1425	640	185	2250
Class (LSTM)	Low congestion	Mild congestion	High congestion	Total
Number of predictions	1605	630	15	2250
Class (DGCN)	Low congestion	Mild congestion	High congestion	Total
Number of predictions	2250	0	0	2250

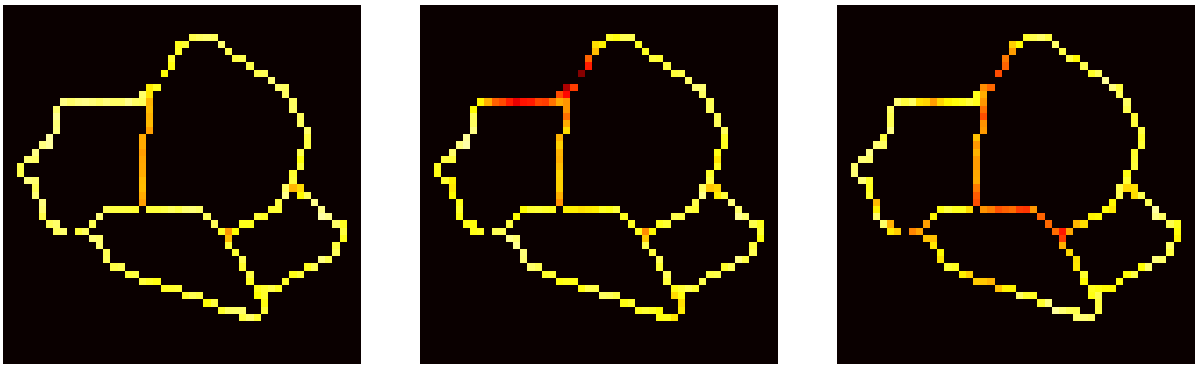
In terms of the congestion level, we can see that the hybrid CNN-RNN model gives the least realistic prediction results with no high congestion level prediction at all. Figure 5.26 shows the DGCN predicted examples from the 3 congestion levels.



(a) Class 0, low congestion (b) Class 1, mild congestion (c) Class 2, high congestion

Figure 5.26: Examples of DGCN predictions belonging to 3 congestion levels

In terms of RMSE, it is clear that DGCN predicts with the lowest error, followed by LSTM encoder-decoder and hybrid CNN-RNN model. Figure 5.27 shows the hybrid model predicted examples from the 3 error classes.

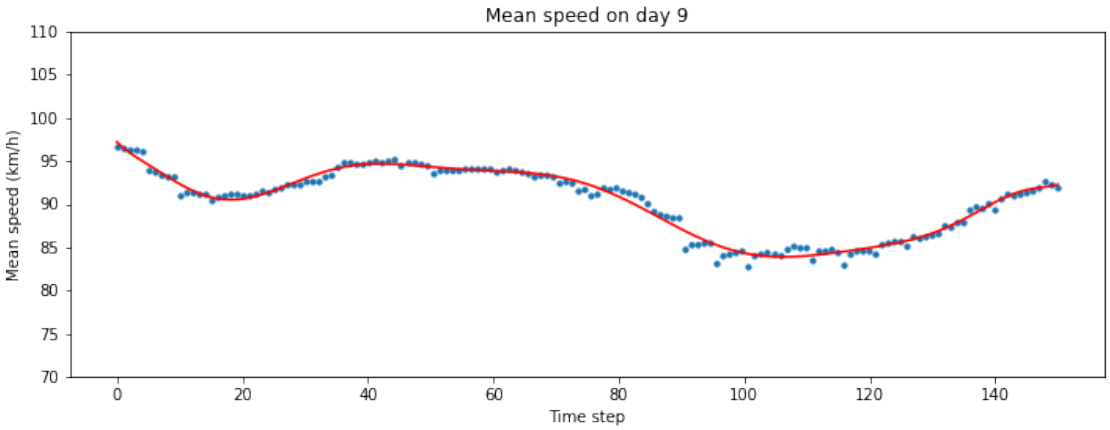


(a) Class 0, low prediction error (b) Class 1, medium prediction error (c) Class 2, high prediction error

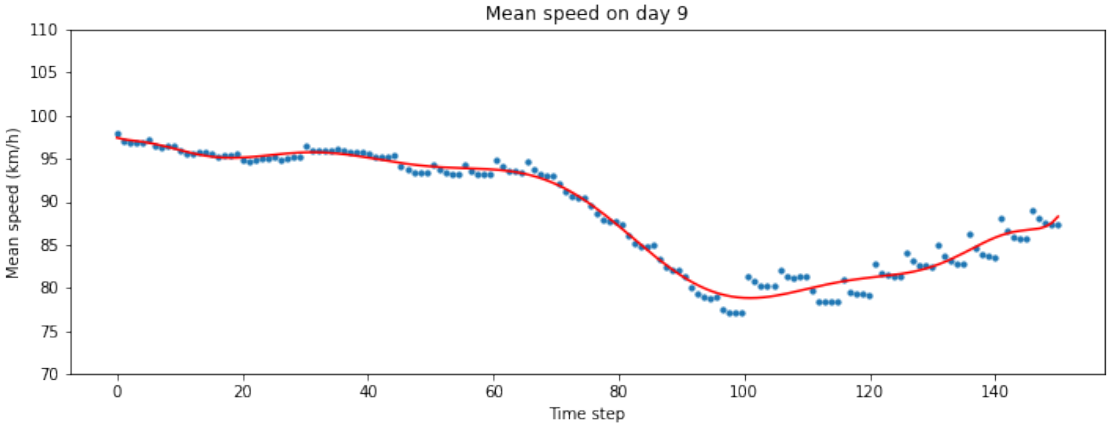
Figure 5.27: Examples of hybrid model predictions belonging to 3 error classes

Figure 5.28 and 5.29 give examples of the distributions in one day of aggregated mean speed and prediction error used to label the prediction images. The distributions of all 15 predicted days are placed in Appendix A Figure A.1-A.6. It can be observed that at around time steps 20 and 100, the network is congested; and the corresponding prediction errors are higher than other time steps. This observation is in line with Figure 5.22, manifesting the prediction unreliability on congestion.

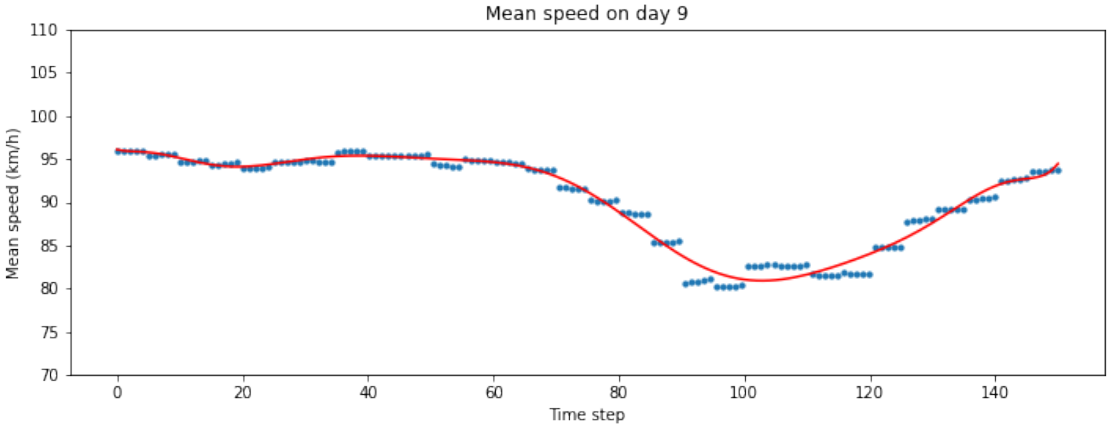
It can be easily observed from Table 5.8 and 5.9 that the samples have an extremely imbalanced distribution. Therefore, during the training of the classifier, class weights are computed and added to the model. However, the model shows no improvement during the training process on both datasets, namely the classifier training was a failure. As the images were properly normalized by keras.preprocess_input, the problem might be because the complex model is overfitting the training data.



(a) Hybrid model aggregated prediction error

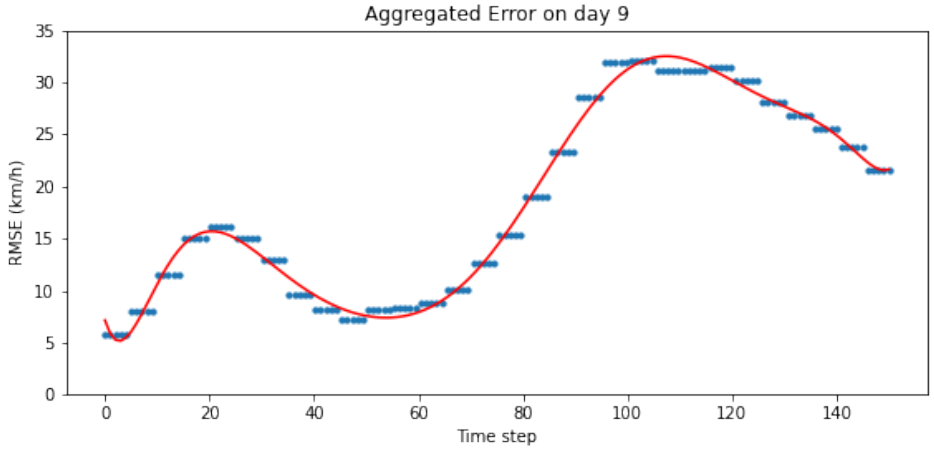


(b) LSTM encoder-decoder aggregated prediction error

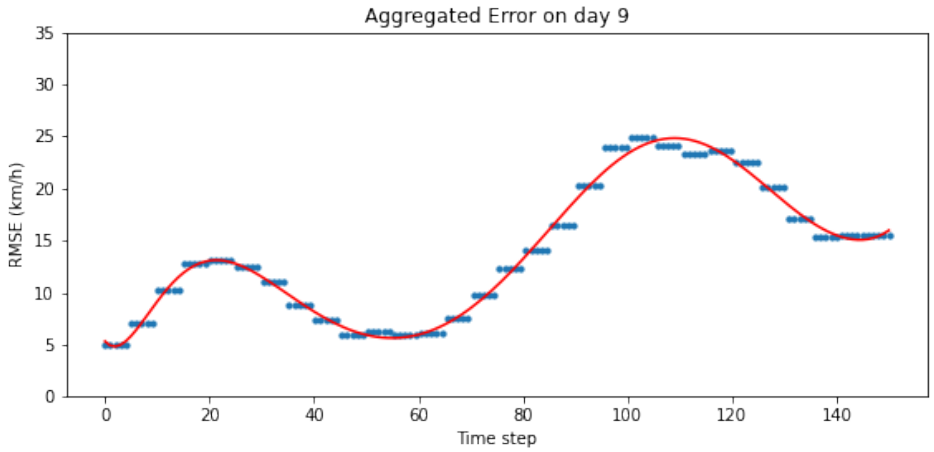


(c) DGCN aggregated prediction error

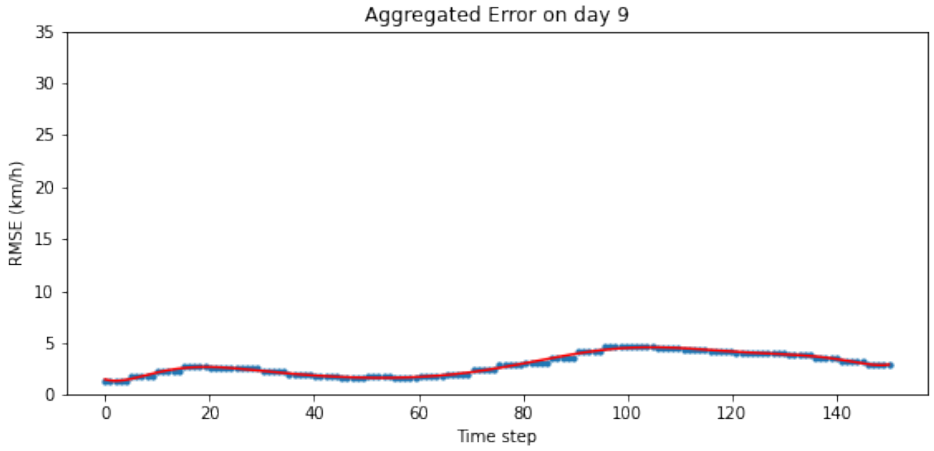
Figure 5.28: Aggregated mean speed on day 9 of the 3 models



(a) Hybrid model aggregated prediction error



(b) LSTM encoder-decoder aggregated prediction error



(c) DGCN aggregated prediction error

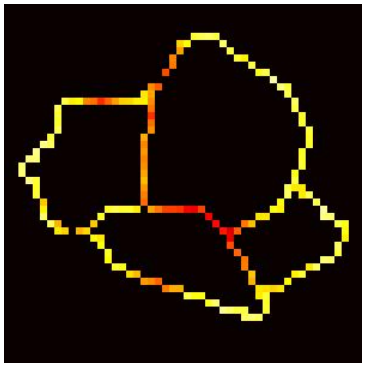
Figure 5.29: Aggregated prediction error on day 9 of the 3 models

5.6. Network Explanation

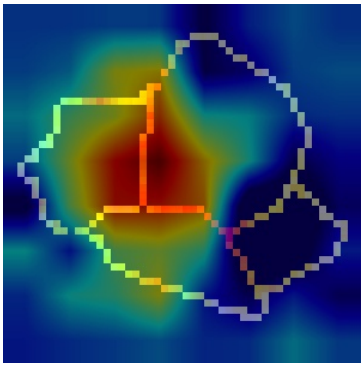
Although the classification model failed to identify the distinct traffic patterns, the Grad-CAMs obtained can still be observed and discussed. Below are examples of Grad-CAMs for the congestion level classification (Figure 5.30) and error level classification (Figure 5.31) of a prediction sequence.

When classifying the congestion level, it seems that the classifier model takes A10 west as the most important region for class 1: mild congestion. While for class 0: low congestion, other areas in the image are highlighted. It's intuitive that the heatmap highlights the area on the road network, and covers the area that has the relatively larger change of values as Grad-CAM is a gradient-based method.

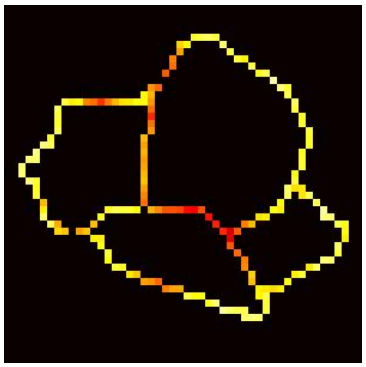
Observing Figure 5.31g - 5.31l, we can see that the heatmap expanded to newly predicted congestion. Hence, combining the findings above, it can be concluded that the hybrid CNN-RNN and LSTM encoder-decoder proposed in this project have the ability to detect congestion and predict speed multi steps ahead based on historical data. However, they are weak in identifying the underlying patterns and development of traffic, namely, they cannot fully capture the dynamic characteristics of the traffic, nor can they accurately provide predictions. Of course, these can be caused by the lack of fine-tuning and rigorous experiments or the incompatibility between the model and the data.



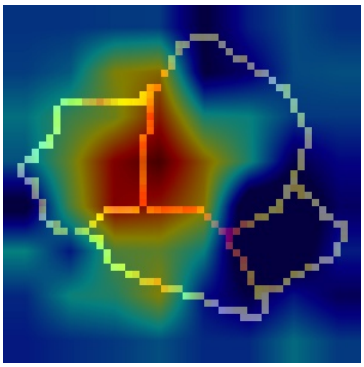
(a) Hybrid model prediction
day 9 time step 90



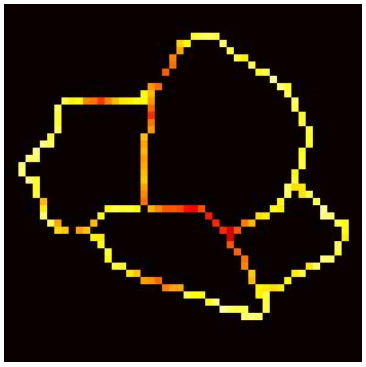
(b) Grad-CAM
day 9 time step 90



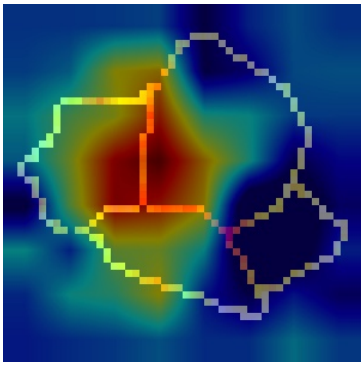
(c) Hybrid model prediction
day 9 time step 91



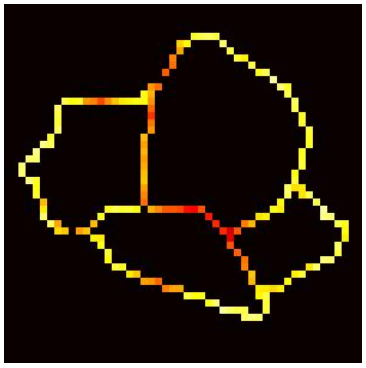
(d) Grad-CAM
day 9 time step 91



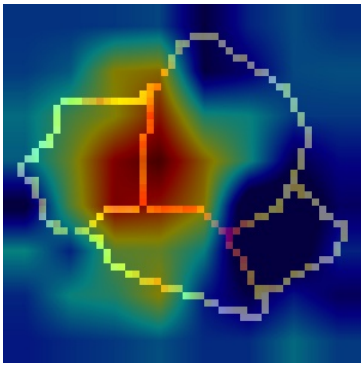
(e) Hybrid model prediction
day 9 time step 92



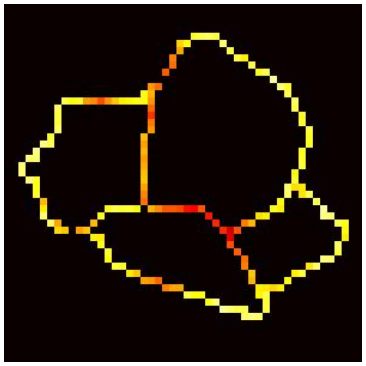
(f) Grad-CAM
day 9 time step 92



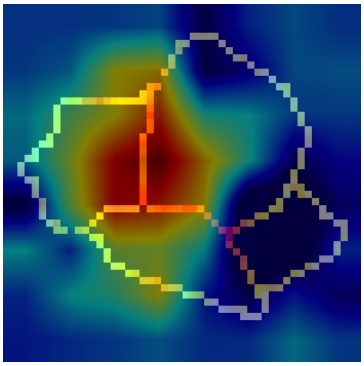
(g) Hybrid model prediction
day 9 time step 93



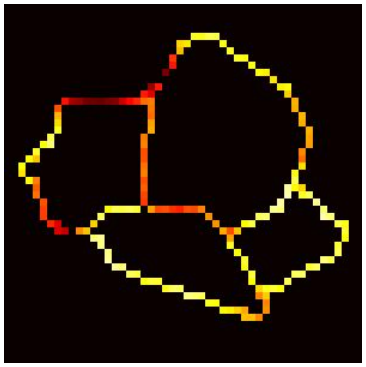
(h) Grad-CAM
day 9 time step 93



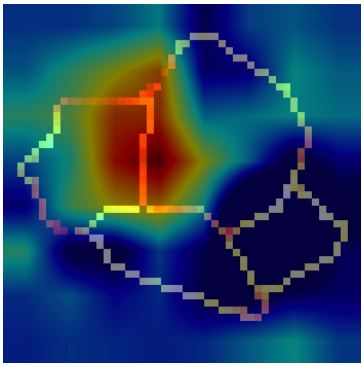
(i) Hybrid model prediction
day 9 time step 94



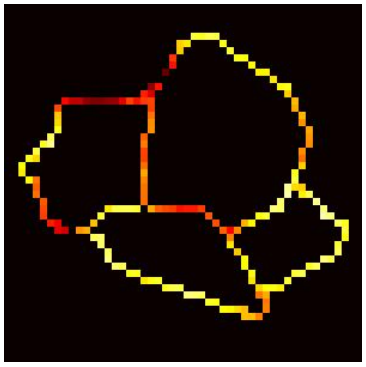
(j) Grad-CAM
day 9 time step 94



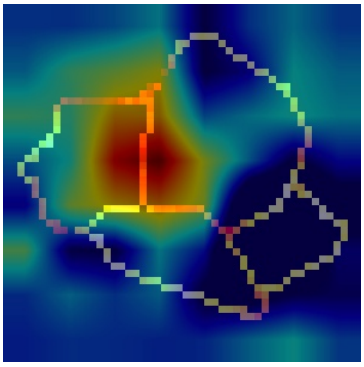
(k) Hybrid model prediction
day 9 time step 95



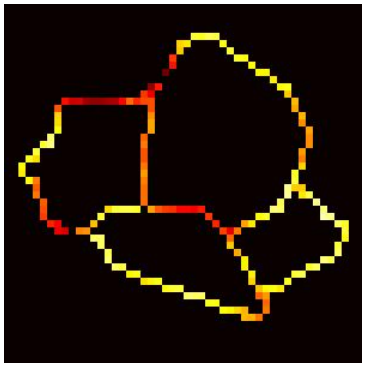
(l) Grad-CAM
day 9 time step 95



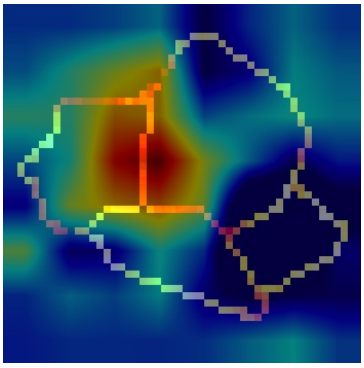
(m) Hybrid model prediction
day 9 time step 96



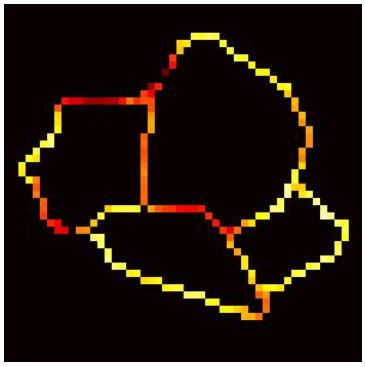
(n) Grad-CAM
day 9 time step 96



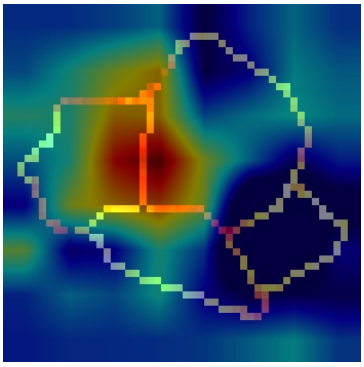
(o) Hybrid model prediction
day 9 time step 97



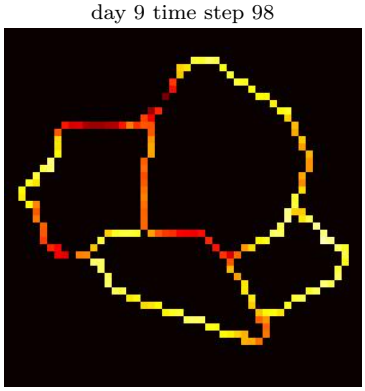
(p) Grad-CAM
day 9 time step 97



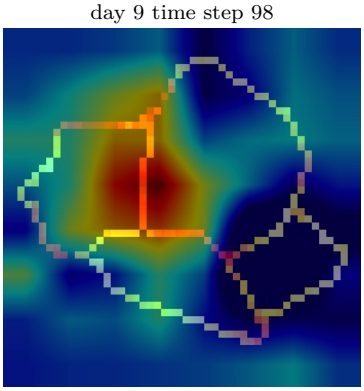
(q) Hybrid model prediction
day 9 time step 98



(r) Grad-CAM
day 9 time step 98

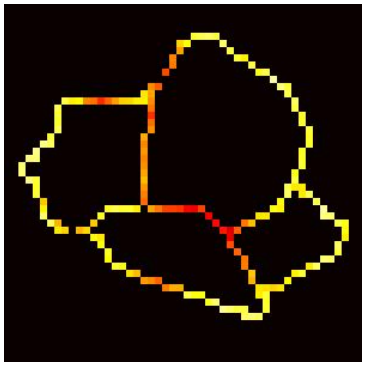


(s) Hybrid model prediction
day 9 time step 99

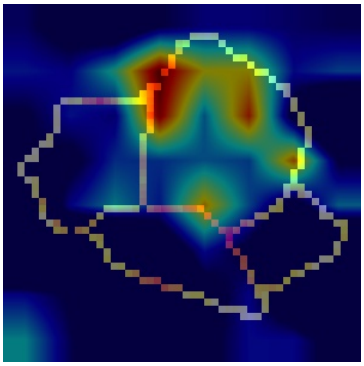


(t) Grad-CAM
day 9 time step 99

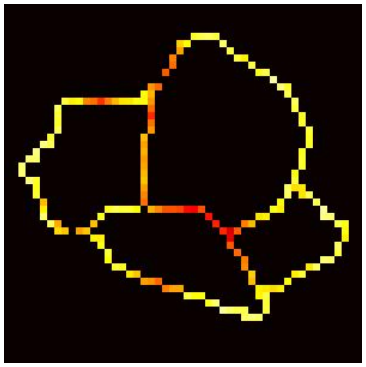
Figure 5.30: Predictions of hybrid model (left) and congestion classification Grad-CAMs (right) on day 9, time step 90-99



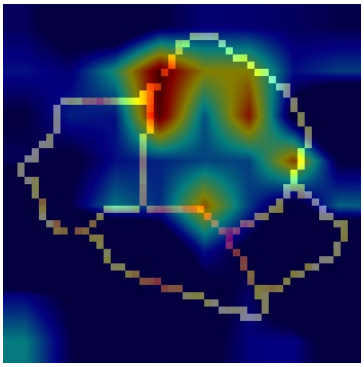
(a) Hybrid model prediction
day 9 time step 90



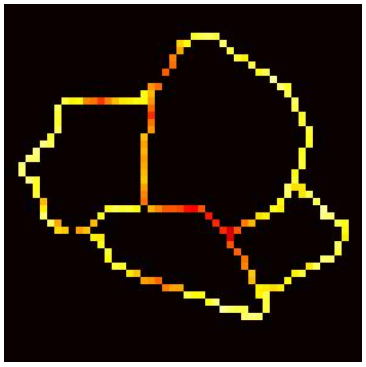
(b) Grad-CAM
day 9 time step 90



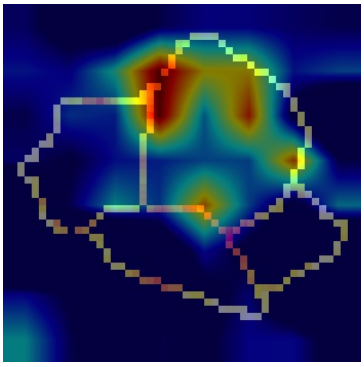
(c) Hybrid model prediction
day 9 time step 91



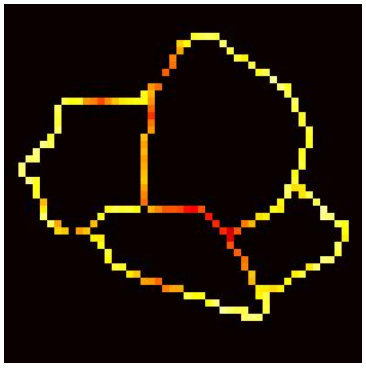
(d) Grad-CAM
day 9 time step 91



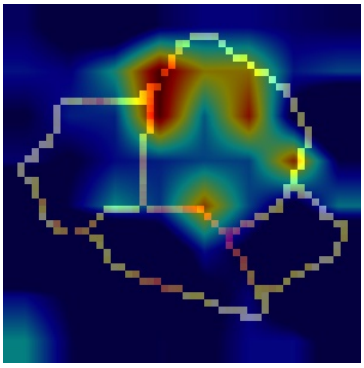
(e) Hybrid model prediction
day 9 time step 92



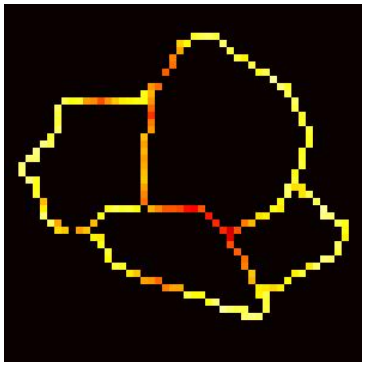
(f) Grad-CAM
day 9 time step 92



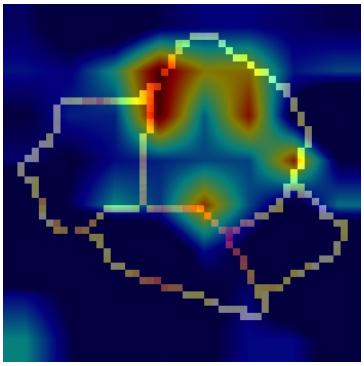
(g) Hybrid model prediction
day 9 time step 93



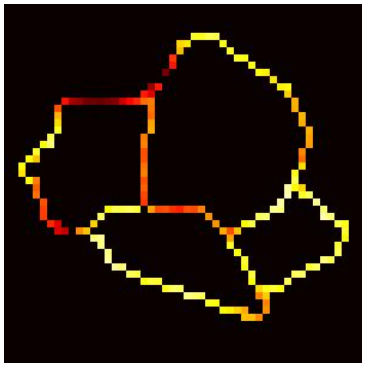
(h) Grad-CAM
day 9 time step 93



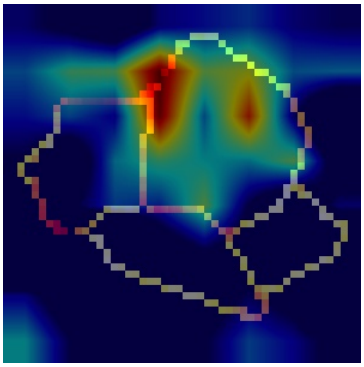
(i) Hybrid model prediction
day 9 time step 94



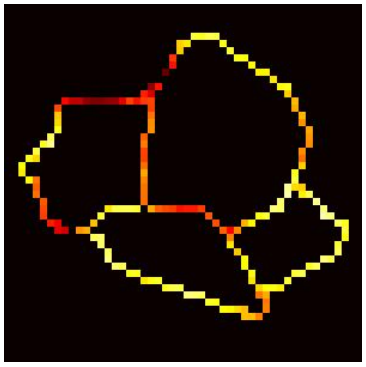
(j) Grad-CAM
day 9 time step 94



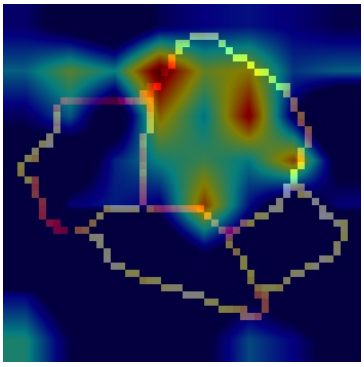
(k) Hybrid model prediction
day 9 time step 95



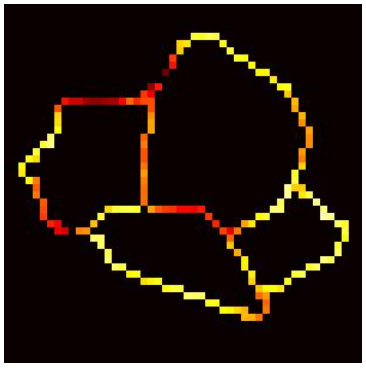
(l) Grad-CAM
day 9 time step 95



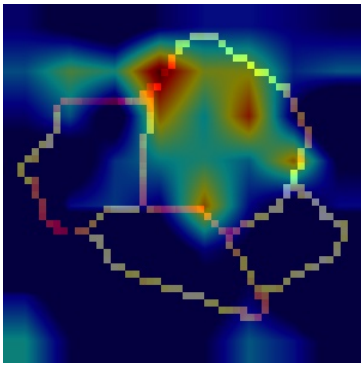
(m) Hybrid model prediction
day 9 time step 96



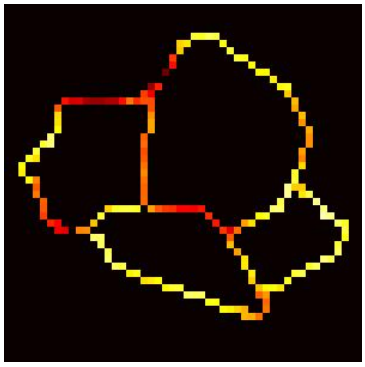
(n) Grad-CAM
day 9 time step 96



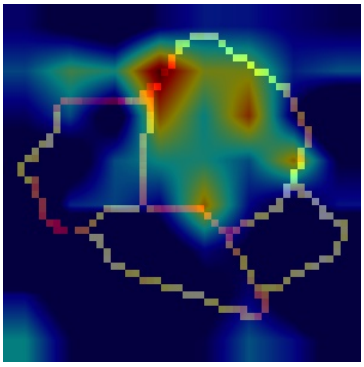
(o) Hybrid model prediction
day 9 time step 97



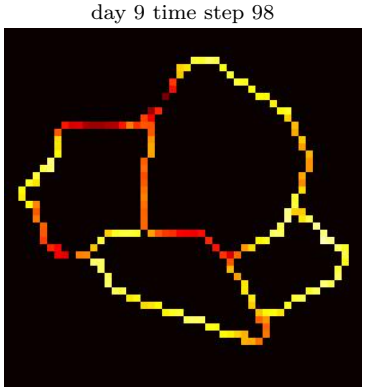
(p) Grad-CAM
day 9 time step 97



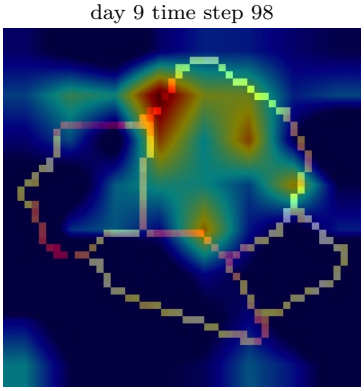
(q) Hybrid model prediction
day 9 time step 98



(r) Grad-CAM
day 9 time step 98



(s) Hybrid model prediction
day 9 time step 99



(t) Grad-CAM
day 9 time step 99

Figure 5.31: Predictions of hybrid model (left) and error classification Grad-CAMs (right) on day 9, time step 90-99

6

Conclusion

This research develops a deep-learning model employing CNN and RNN for network traffic speed prediction, and a CNN-based classification model for traffic pattern identification. Grad-CAM is utilized to generate and visualize explainable classification results, helping explore how deep-learning models identify traffic patterns. A case study is conducted on a realistic freeway network for analysis and verification.

In Section 6.1, the key findings of this research are given, along with the answering of research questions. Section 6.2 summarizes the contributions of this research. The limitations and corresponding recommendations for the future can be found in Section 6.3.

6.1. Key Findings

The hybrid model and LSTM encoder-decoder proposed in this project can detect congestion and predict speed multi steps ahead based on historical data. But they are weak in identifying the underlying patterns and development of traffic, namely, they cannot fully capture the dynamic characteristics of the traffic, nor can they accurately provide predictions. Of course, these can be caused by the lack of fine-tuning and rigorous experiments or the incompatibility between the model and the data.

The unsatisfying performance observed with the use of the Inception ResNet v2 model pretrained on ImageNet for transfer learning, however, does not indicate the failure of this approach for the speed prediction problem, nor the model generalizability. Instead, the author suggests that this outcome highlights the limitations imposed on deep learning models by the training dataset, as known as the inductive bias.

While ImageNet is a vast dataset containing a considerable number of real-world objects, it does not offer a sufficient number of more abstract images, such as the rasterized speed images employed in this research. Thus, the model naturally feels strange to these images and cannot predict/identify distinct patterns very well. Another sign of inductive bias is in the design of Inception ResNet v2, which requires a specific input size, leading to potential weakness regarding data padding/compressing.

Grad-CAMs indicate that the highlighted areas are mostly locations with rapid changes of values, due to the method's gradient-based characteristics. However, as the prediction and classification all have unsatisfying performances, both the quality of data and the compatibility

between the image data and the classifier are low, resulting in the unreliability of Grad-CAM explanations.

So, how do deep neural networks identify spatio-temporal traffic patterns for network-wide traffic predictions?

DNN reads historical data and learns temporal variations of traffic data, which makes it able to predict future traffic states; it learns spatial characteristics in a somehow numerical way by observing and remembering how the values at specific areas change and mimicking this change for future predictions aiming for the smallest loss. Without driving directions encoded in the data, computer-vision-based DNN actually fails to learn this from static images. By finding similarities between images in the same class, DNN attempts to identify the similarities as the underlying patterns, yet this identification rationale is not as expected.

- What are the state-of-the-art deep learning methods used for network-wide traffic prediction?

  The state-of-the-art deep learning methods used for network-wide traffic prediction include but are not limited to convolutional neural networks (CNNs), recurrent neural networks (RNNs) (especially long short-time memory neural networks), attention-based neural networks, and graph-based neural networks. In contrast to conventional methods, deep learning models have the ability to handle high-dimensional and heterogeneous data inputs, including spatio-temporal traffic data, satellite imagery, and weather data. The ability of deep learning models, such as CNNs and RNNs, to detect spatio-temporal patterns in traffic data permits them to capture the interactions and dependencies between different road segments and their influence on traffic speeds. Moreover, deep learning models can be trained end-to-end, which eliminates the requirement for hand-engineered features and permits them to handle large quantities of data with ease.

- How to build a computer vision-based network-wide traffic prediction model using deep learning methods?

  Computer vision methods involve a wide range of techniques, including image processing, feature extraction, object detection and recognition, segmentation, and classification. The key is to employ models that are capable of processing and learning images, thus CNNs are classic suitable choices. In this project, an Inception ResNet v2-based deep learning model is constructed, with the Inception ResNet v2 as a feature extractor and an LSTM encoder-decoder stacking on top of it. This way, the extracted features are fed into the encoder-decoder, and the model will learn temporal characteristics and predict future features. Using transposed convolution, the predicted features can be transformed back to images, so that both the spatial and temporal features are taken into account.

- Which spatio-temporal properties of traffic contribute to identifying distinct traffic patterns using DNN?

  Although the image classifier fails to identify distinct traffic patterns, speeds and the generation and propagation of congestion contribute to the identification, which can be concluded from the obtained Grad-CAMs. DNN, especially computer vision-based DNN,

largely focuses on the edges, curves, and parts in images. In rasterized speed images, the edges are the road networks; and the parts with a rapid change of values are the congestion wave and moving jams, which are all highlighted in Grad-CAMs.

- How to relate traffic patterns to the features extracted using DNN?

  Heatmaps or other visualization approaches help. Feature importance analysis, such as gradient-based attribution methods, can also help identify which features have the most influence on the model's predictions. In this project, Grad-CAM is used to help relate traffic patterns to extracted features. The DNN proposed in this project extracts features through convolutional layers. By obtaining weighted activation maps using the gradients, intuitive class activation maps can be generated and used to find a potential relationship between traffic patterns and extracted features. The highlighted features are the features that contribute to the result of a certain class; the class can be a representation of a specific traffic pattern.

6.2. Contributions

The scientific contributions of this study are as follows:

- A hybrid CNN-RNN model utilizing a pretrained Inception ResNet v2 feature extractor and an LSTM encoder-decoder is constructed to forecast network traffic speeds with the objective of capturing spatio-temporal characteristics of the traffic.
- Based on the image predictions, a deep learning-based classification model including the convolution layers of pretrained Inception ResNet v2 and 2 self-stacked layers is developed to identify underlying distinct traffic patterns.
- To address the unexplainability of the constructed classifier, an attention-based model explaining approach, Grad-CAM, is employed to generate explainable classification results and visualize what the DNN is learning during detecting and identifying traffic patterns.
- A real freeway network is taken as a case study to verify the method.

And the practical contributions include:

- The developed model, upon fine-tuning, can be used by authorities as a powerful tool assisting policy making. It can also help travelers make more informed decisions about their travel plans and routes.
- The data rasterization method is considered suitable for visualizing the real-time traffic conditions on the road network, which can be applied to electric variable traffic sign boards.
- An experiment using transfer learning is conducted to explore the possibility of utilizing pretrained models on traffic speed prediction problems. Authorities or companies that have insufficient traffic data or unadvanced equipment can still implement deep learning techniques, as long as there is a proper pretrained model available.
- Grad-CAM provides a way to visualize and interpret the predictions made by the CNN classifier, making it easier to understand how the model is making its predictions.

6.3. Limitation & Recommendation

There are some limitations in this project, which would be potential future research directions:

- In terms of data rasterization, although the spatial structure was preserved, the detailed information like driving direction was not able to be encoded, leading to the failure of learning congestion propagation for the hybrid CNN-RNN model and LSTM encoder-decoder. Further research can explore better approaches to encode traffic data and keep important information.
- The size of initial rasterized images was (50, 50), which were further resized up to (299, 299). A (50, 50) image contains 2500 pixels, and only 181 of them represent the road network, which is too sparse for DNN to learn. Further studies should fully consider the network input requirement and data characteristics so that a suitable model can be chosen for the prediction task.
- The input images are all in one direction, leading to the lack of further testing on CNN's ability to learn spatial characteristics. By flipping and rotating the images, another input dataset can be generated and employed to compare the predictions given by the models, thus gaining insights into to what extent the models identify spatial patterns.
- As time is limited, only 1 road network is used as a case study. Further works can try to evaluate the models under different situations to further assess model generalizability and scalability, e.g. different network scales, and different network resolutions.
- In this project, the author used speed to predict speed. While traffic flow is also an essential traffic variable, a possible attempt could be to predict traffic speed with both speed and flow data.
- Due to the equipment limitation, no fine-tuning or grid searching was performed during the project. And only 1 set of parameters (learning rate, decay rate, training epochs, and batch size) was used. Hence, the results are less reliable. If allowed, further research can tune the models and compare them with their best performance.
- There can be more in-depth explorations regarding the effects of different parameter values and data processing approaches on the model performance. For example, conducting experiments for 5-step and 1-step sliding windows respectively to check possible influences can be interesting to include.
- The robustness of the model was not tested. If random simulated incidents are artificially added to the test data, the potential difference between random incident prediction and original test prediction can be explored to see whether the model truly learns traffic.
- Grad-CAM is a semi-interpreting approach, which only visualizes the output of convolutional layers but has no explanations for the mechanism. Further studies can try to analyze and decompose deep neural networks for more theoretical explanations.

References

Abdelraouf, A., Abdel-Aty, M., & Yuan, J. (2022). Utilizing attention-based multi-encoder-decoder neural networks for freeway traffic speed prediction. IEEE Transactions on Intelligent Transportation Systems, 23(8), 11960–11969. https://doi.org/10.1109/TITS.2021.3108939

Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H., & Mishalani, R. (2000). Dynamit: A simulation-based system for traffic prediction.

Caliper. (1996). Transcad transportation planning software 3.0. Retrieved April 1, 2022, from https://www.caliper.com/tcovu.htm

Castro, J., Mantas, C., & Benitez, J. (2002). Interpretation of artificial neural networks by means of fuzzy rules. IEEE Transactions on Neural Networks, 13(1), 101–116. https://doi.org/10.1109/72.977279

Chai, D., Wang, L., & Yang, Q. (2018). Bike flow prediction with multi-graph convolutional networks. Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems, 397–400.

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 839–847. https://doi.org/10.1109/WACV.2018.00097

Cheng, X., Zhang, R., Zhou, J., & Xu, W. (2018). Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. 2018 International Joint Conference on Neural Networks (IJCNN), 1–8. https://doi.org/10.1109/IJCNN.2018.8489600

Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143.

Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K., & Das, P. (2018). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. https://doi.org/10.48550/ARXIV.1802.07623

Domencich, T. A., & McFadden, D. (1975). Urban travel demand-a behavioral analysis (tech. rep.).

Du, S., Li, T., Gong, X., Yang, Y., & Horng, S. J. (2017). Traffic flow forecasting based on hybrid deep learning framework. 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 1–6. https://doi.org/10.1109/ISKE.2017.8258813

Erlander, S., & Stewart, N. F. (1990). The gravity model in transportation analysis: Theory and extensions (Vol. 3). Vsp.

Fang, M., Yin, J., Zhu, X., & Zhang, C. (2015). Trgraph: Cross-network transfer learning via common signature subgraphs. IEEE Transactions on Knowledge and Data Engineering, 27(9), 2536–2549. https://doi.org/10.1109/TKDE.2015.2413789

Fouladgar, M., Parchami, M., Elmasri, R., & Ghaderi, A. (2017). Scalable deep traffic flow neural networks for urban traffic congestion prediction. 2017 International Joint Conference on Neural Networks (IJCNN), 2251–2258. https://doi.org/10.1109/IJCNN.2017.7966128

Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. Proceedings of the AAAI conference on artificial intelligence, 33(01), 922–929.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review [Recent Developments on Deep Big Vision]. Neurocomputing, 187, 27–48. https://doi.org/https://doi.org/10.1016/j.neucom.2015.09.116

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. https://doi.org/10.48550/ARXIV.1512.03385

Huang, W., Song, G., Hong, H., & Xie, K. (2014). Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. IEEE Transactions on Intelligent Transportation Systems, 15(5), 2191–2201. https://doi.org/10.1109/TITS.2014.2311123

Jia, Y., Wu, J., & Du, Y. (2016). Traffic speed prediction using deep learning method. 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 1217–1222. https://doi.org/10.1109/ITSC.2016.7795712

Jiang, P.-T., Zhang, C.-B., Hou, Q., Cheng, M.-M., & Wei, Y. (2021). Layercam: Exploring hierarchical class activation maps for localization. IEEE Transactions on Image Processing, 30, 5875–5888. https://doi.org/10.1109/TIP.2021.3089943

Jin, X., Zhang, Y., & Yao, D. (2007). Simultaneously prediction of network traffic flow based on pca-svr. In D. Liu, S. Fei, Z. Hou, H. Zhang, & C. Sun (Eds.), Advances in neural networks – isnn 2007 (pp. 1022–1031). Springer Berlin Heidelberg.

Karim, M. M., Li, Y., & Qin, R. (2022). Toward explainable artificial intelligence for early anticipation of traffic accidents. Transportation research record, 2676(6), 743–755.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kitamura, R., Chen, C., Pendyala, R. M., & Narayanan, R. (2000). Micro-simulation of daily activity-travel patterns for travel demand forecasting. Transportation, 27(1), 25–51.

Krishnakumari, P., Perotti, A., Pinto, V., Cats, O., & van Lint, H. (2018). Understanding network traffic states using transfer learning. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 1396–1401.

Kuang, L., Yan, X., Tan, X., Li, S., & Yang, X. (2019). Predicting taxi demand based on 3d convolutional neural network and multi-task learning. Remote Sensing, 11(11), 1265.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

Li, G., Knoop, V. L., & van Lint, H. (2021). Multistep traffic forecasting by dynamic graph convolution: Interpretations of real-time spatial correlations. Transportation Research Part C: Emerging Technologies, 128, 103185. https://doi.org/https://doi.org/10.1016/j.trc.2021.103185

Liang, Y., Ouyang, K., Jing, L., Ruan, S., Liu, Y., Zhang, J., Rosenblum, D. S., & Zheng, Y. (2019). Urbanfm: Inferring fine-grained urban flows. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 3132–3142. https://doi.org/10.1145/3292500.3330646

Liao, B., Zhang, J., Wu, C., McIlwraith, D., Chen, T., Yang, S., Guo, Y., & Wu, F. (2018). Deep sequence learning with auxiliary information for traffic prediction. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 537–546.

Liu, Z., Li, Z., Wu, K., & Li, M. (2018). Urban traffic prediction from mobility data using deep learning. IEEE Network, 32(4), 40–46. https://doi.org/10.1109/MNET.2018.1700411

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F.-Y. (2015). Traffic flow prediction with big data: A deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865–873. https://doi.org/10.1109/TITS.2014.2345663

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., & Wang, Y. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. Sensors, 17(4). https://doi.org/10.3390/s17040818

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54, 187–197. https://doi.org/https://doi.org/10.1016/j.trc.2015.03.014

Maree, C., & Omlin, C. W. (2022). Understanding spending behavior: Recurrent neural network explanation and interpretation. 2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr), 1–7.

Omeiza, D., Speakman, S., Cintas, C., & Weldemariam, K. (2019). Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. CoRR, abs/1908.01224.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345–1359.

Rahmani, M., Jenelius, E., & Koutsopoulos, H. N. (2013). Route travel time estimation using low-frequency floating car data. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2292–2297. https://doi.org/10.1109/ITSC.2013.6728569

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 1135–1144.

Rosario, G., Sonderman, T., & Zhu, X. (2018). Deep transfer learning for traffic sign recognition. 2018 IEEE International Conference on Information Reuse and Integration (IRI), 178–185. https://doi.org/10.1109/IRI.2018.00034

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (tech. rep.). California Univ San Diego La Jolla Inst for Cognitive Science.

Saadallah, A., Moreira-Matias, L., Sousa, R., Khiari, J., Jenelius, E., & Gama, J. (2020). Bright—drift-aware demand predictions for taxi networks. IEEE Transactions on Knowledge and Data Engineering, 32(2), 234–245. https://doi.org/10.1109/TKDE.2018.2883616

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. Proceedings of the IEEE international conference on computer vision, 618–626.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: Removing noise by adding noise. arXiv preprint arXiv:1706.03825.

Sohn, K. (2020). Forecasting road traffic speeds by considering area-wide spatiotemporal dependencies based on a graph convolutional neural network (gcn). Transportation Research Part C Emerging Technologies, 114, 189–204. https://doi.org/10.1016/j.trc.2020.02.013

Sun, S., Chen, J., & Sun, J. (2019). Traffic congestion prediction based on gps trajectory data. International Journal of Distributed Sensor Networks, 15(5), 1550147719847440. https://doi.org/10.1177/1550147719847440

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. https://doi.org/10.48550/ARXIV.1602.07261

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. https://doi.org/10.48550/ARXIV.1409.4842

Tajaddini, A., Rose, G., Kockelman, K. M., & Vu, H. L. (2020). Recent progress in activity-based travel demand modeling: Rising data and applicability. In S. de Luca, R. D. Pace, & C. Fiori (Eds.), Models and technologies for smart, sustainable and safe transportation systems. IntechOpen. https://doi.org/10.5772/intechopen.93827

Tang, J., Chen, X., Hu, Z., Zong, F., Han, C., & Li, L. (2019). Traffic flow prediction based on combination of support vector machine and data denoising schemes. Physica A: Statistical Mechanics and its Applications, 534, 120642.

Tang, J., Gao, F., Liu, F., & Chen, X. (2020). A denoising scheme-based traffic flow prediction model: Combination of ensemble empirical mode decomposition and fuzzy c-means neural network. IEEE Access, 8, 11546–11559.

Tedjopurnomo, D. A., Bao, Z., Zheng, B., Choudhury, F. M., & Qin, A. K. (2022). A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. IEEE Transactions on Knowledge and Data Engineering, 34(4), 1544–1561. https://doi.org/10.1109/TKDE.2020.3001195

Tian, C., & Chan, W. K. (2021). Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies. IET Intelligent Transport Systems, 15(4), 549–561.

Traffic Simulation Software | PTV Vissim | PTV Group. (1992). Ptv vissim. Retrieved April 1, 2022, from https://www.ptvgroup.com/en/solutions/products/ptv-vissim/

Wang, D., Zhang, J., Cao, W., Li, J., & Zheng, Y. (2018). When will you arrive? estimating travel time based on deep neural networks. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

Wang, H., Li, Z., Kuo, Y., & Kifer, D. (2015). A simple baseline for travel time estimation using large-scale trip data. CoRR, abs/1512.08580.

Wang, J., Zhang, W., Yang, H., Yeh, C.-C. M., & Wang, L. (2021). Visual analytics for rnn-based deep reinforcement learning. IEEE Transactions on Visualization and Computer Graphics.

Wang, T. (2019). Gaining free or low-cost interpretability with interpretable partial substitute. International Conference on Machine Learning, 6505–6514.

Wang, W., & Li, X. (2018). Travel speed prediction with a hierarchical convolutional neural network and long short-term memory model framework. CoRR, abs/1809.01887.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3(1), 1–40.

Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. Journal of transportation engineering, 129(6), 664–672.

Wilson, A. (2013). Entropy in urban and regional modelling (routledge revivals). Routledge.

Wu, M., Parbhoo, S., Hughes, M. C., Kindle, R., Celi, L. A., Zazzi, M., Roth, V., & Doshi-Velez, F. (2019). Regional tree regularization for interpretability in black box models. CoRR, abs/1908.04494. http://arxiv.org/abs/1908.04494

Wu, Y., & Tan, H. (2016). Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. CoRR, abs/1612.01022. http://arxiv.org/abs/1612.01022

Yang, B., Guo, C., & Jensen, C. S. (2013). Travel cost inference from sparse, spatio temporally correlated time series using markov models. Proceedings of the VLDB Endowment, 6(9), 769–780.

Yao, H., Tang, X., Wei, H., Zheng, G., & Li, Z. (2019). Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. Proceedings of the AAAI conference on artificial intelligence, 33(01), 5668–5675.

Yeon, K., Min, K., Shin, J., Sunwoo, M., & Han, M. (2019). Ego-vehicle speed prediction using a long short-term memory based recurrent neural network. International Journal of Automotive Technology, 20(4), 713–722.

Yoshida, A., Yatsushiro, Y., Hata, N., Higurashi, T., Tateiwa, N., Wakamatsu, T., Tanaka, A., Nagamatsu, K., & Fujisawa, K. (2019). Practical end-to-end repositioning algorithm for managing bike-sharing system. 2019 IEEE International Conference on Big Data (Big Data), 1251–1258.

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, 3634–3640. https://doi.org/10.24963/ijcai.2018/505

Yu, H., Wu, Z., Wang, S., Wang, Y., & Ma, X. (2017). Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. Sensors, 17(7), 1501.

Zhang, H., Wu, H., Sun, W., & Zheng, B. (2018). Deeptravel: A neural network based travel time estimation model with auxiliary supervision. CoRR, abs/1802.02147.

Zhang, Q., Cao, R., Shi, F., Wu, Y. N., & Zhu, S.-C. (2018). Interpreting cnn knowledge via an explanatory graph. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

Zhang, Q., Wu, Y. N., & Zhu, S.-C. (2018). Interpretable convolutional neural networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8827–8836. https://doi.org/10.1109/CVPR.2018.00920

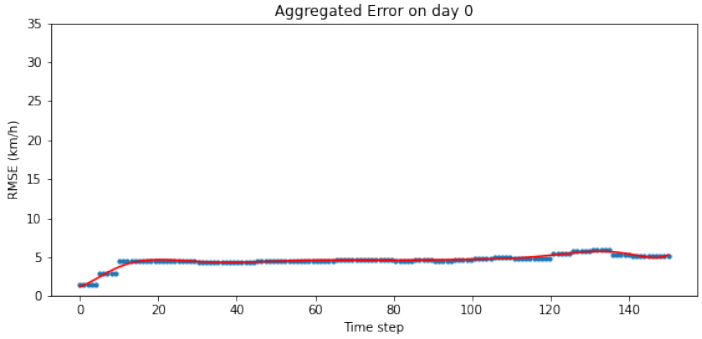
Zhang, Q., Yang, Y., Wu, Y. N., & Zhu, S. (2018). Interpreting cnns via decision trees. CoRR, abs/1802.00121. http://arxiv.org/abs/1802.00121

Zheng, J., & Ni, L. M. (2013). Time-dependent trajectory regression on road networks via multi-task learning. Proceedings of the AAAI Conference on Artificial Intelligence, 27(1), 1048–1055.

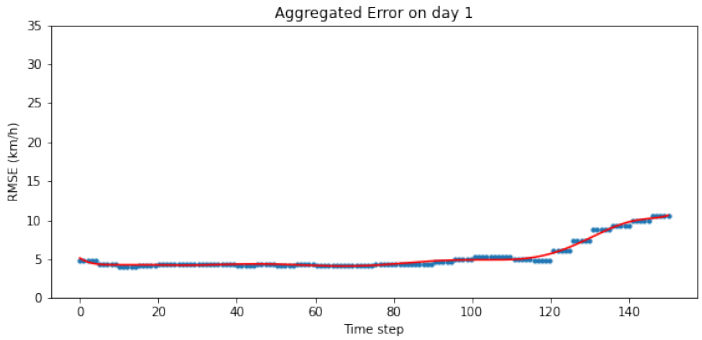
Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2921–2929. https://doi.org/10.1109/CVPR.2016.319

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. Proceedings of the IEEE, 109(1), 43–76.

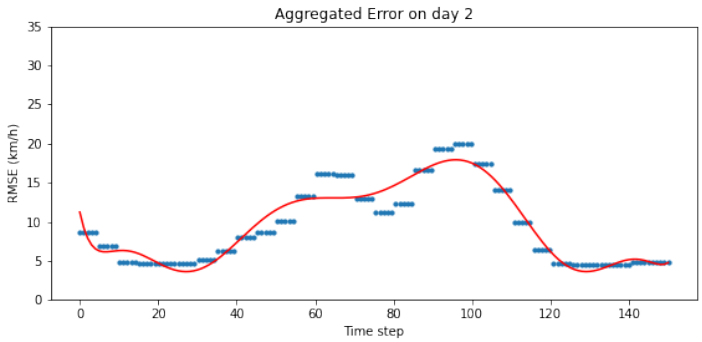
Appendix A



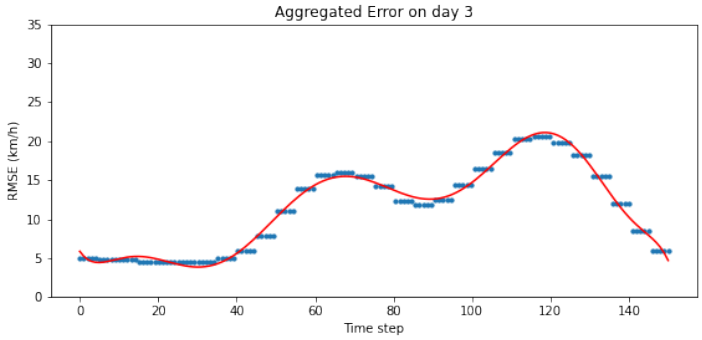
(a)



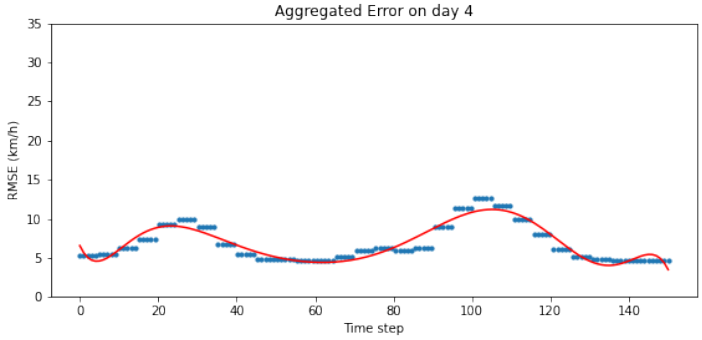
(b)



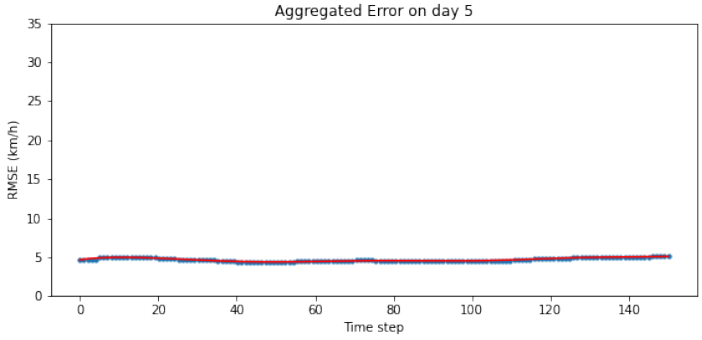
(c)


Aggregated Error on day 3
RMSE (km/h)
Time step

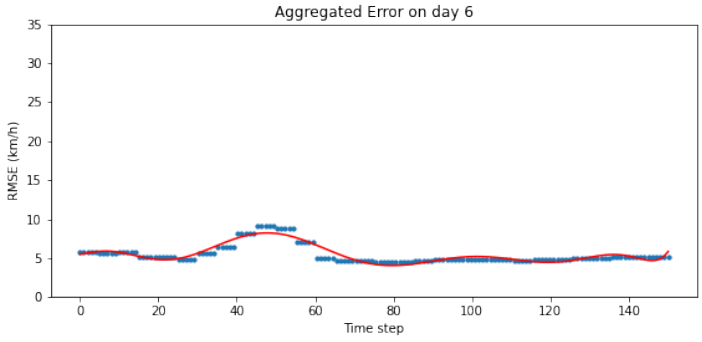
(d)


Aggregated Error on day 4
RMSE (km/h)
Time step

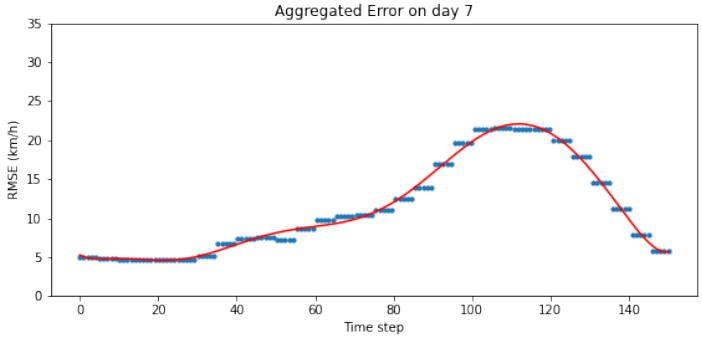
(e)


Aggregated Error on day 5
RMSE (km/h)
Time step

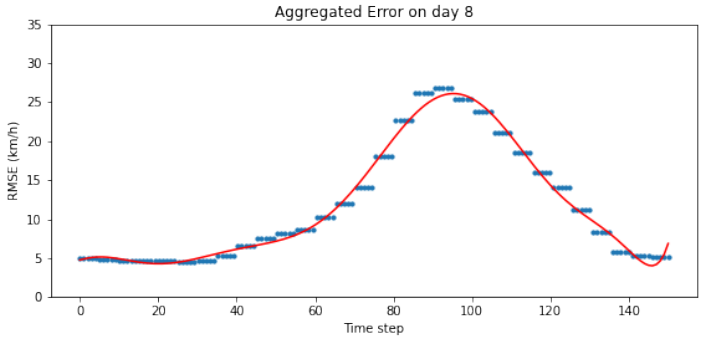
(f)


Aggregated Error on day 6
RMSE (km/h)
Time step

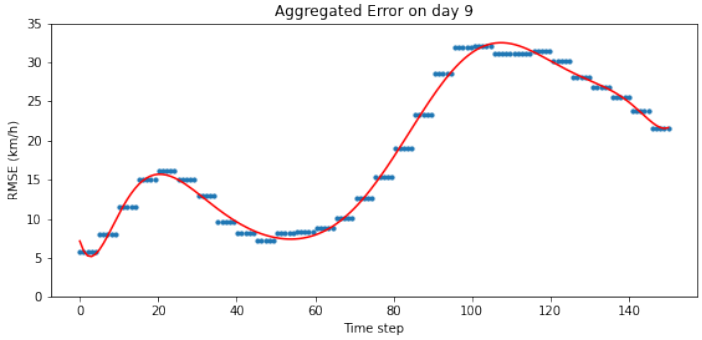
(g)


Aggregated Error on day 7
RMSE (km/h)
Time step

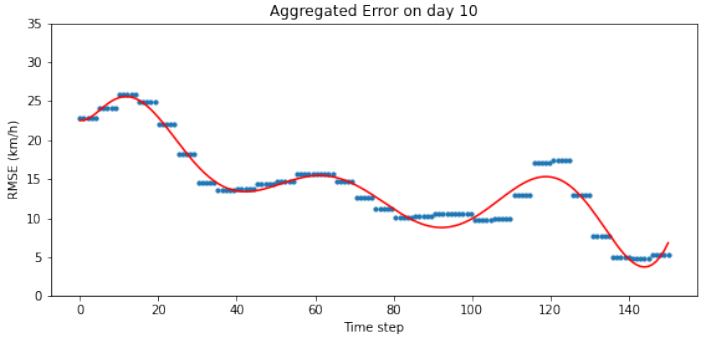
(h)


Aggregated Error on day 8
RMSE (km/h)
Time step

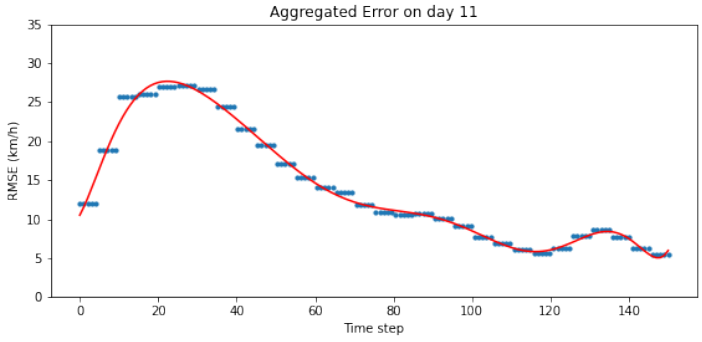
(i)


Aggregated Error on day 9
RMSE (km/h)
Time step

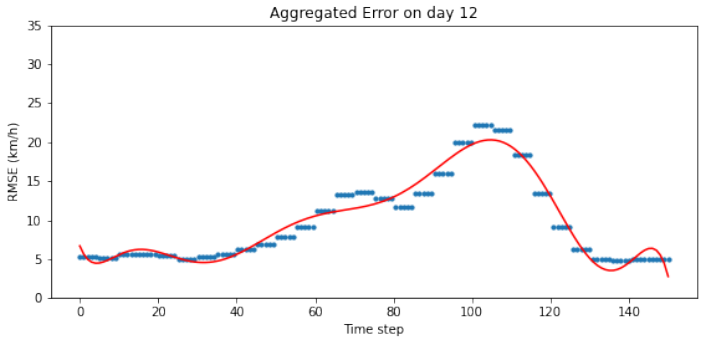
(j)


Aggregated Error on day 10
RMSE (km/h)
Time step

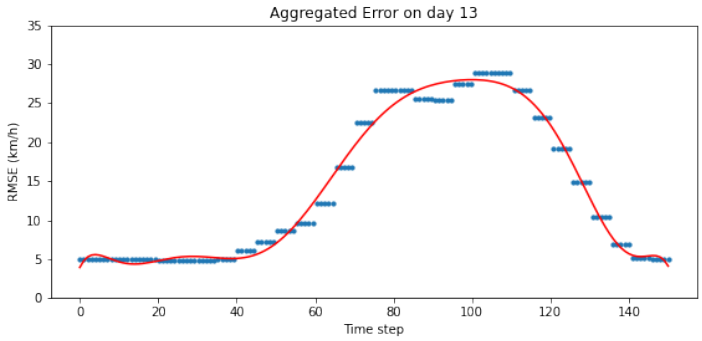
(k)


Aggregated Error on day 11
RMSE (km/h)
Time step

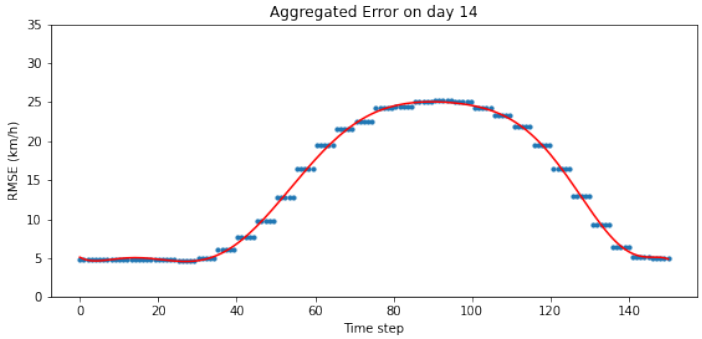
(l)


Aggregated Error on day 12
RMSE (km/h)
Time step

(m)

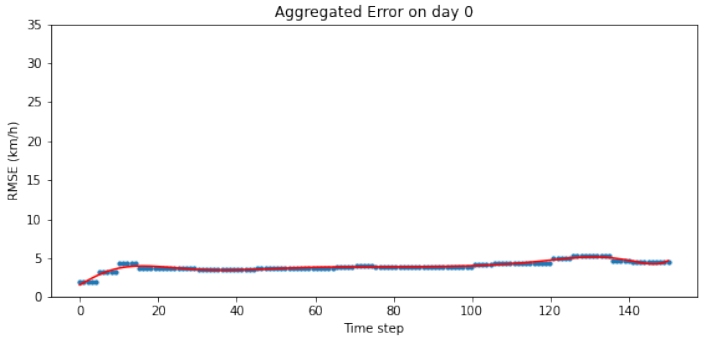


(n)

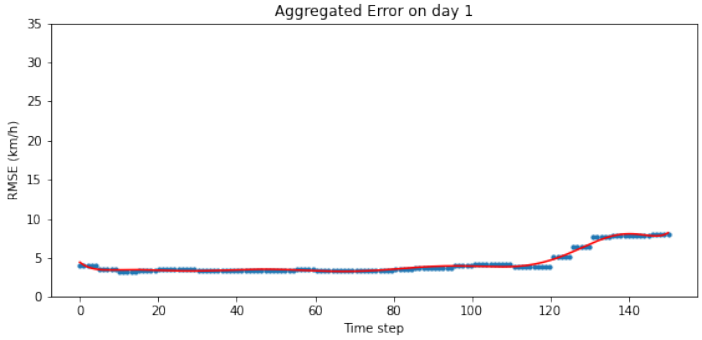


(o)

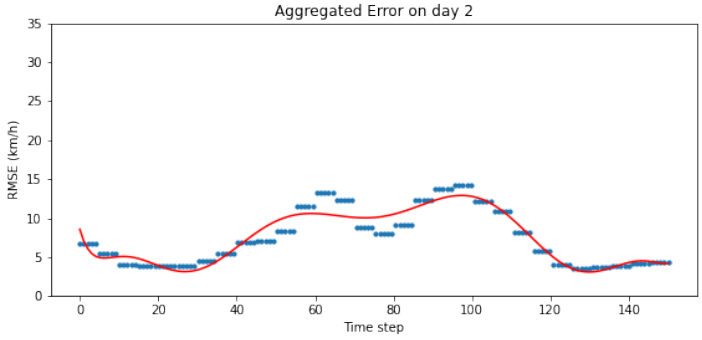
Figure A.1: Aggregated hybrid model prediction error distribution of 15 days


Aggregated Error on day 0
RMSE (km/h)
Time step

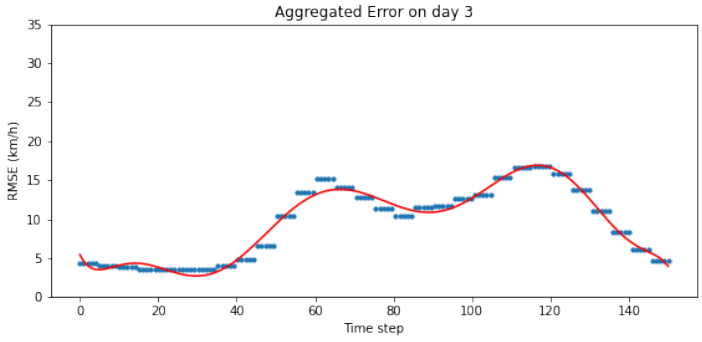
(a)


Aggregated Error on day 1
RMSE (km/h)
Time step

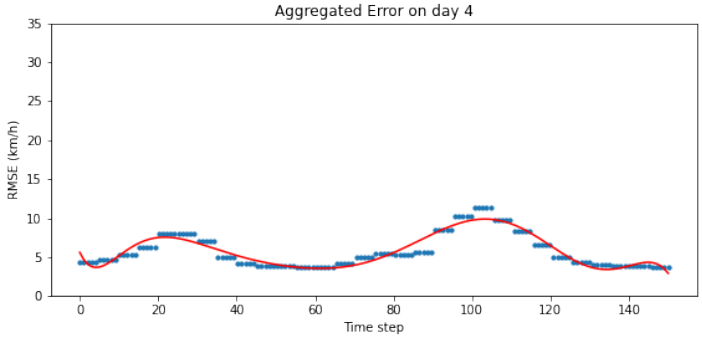
(b)


Aggregated Error on day 2
RMSE (km/h)
Time step

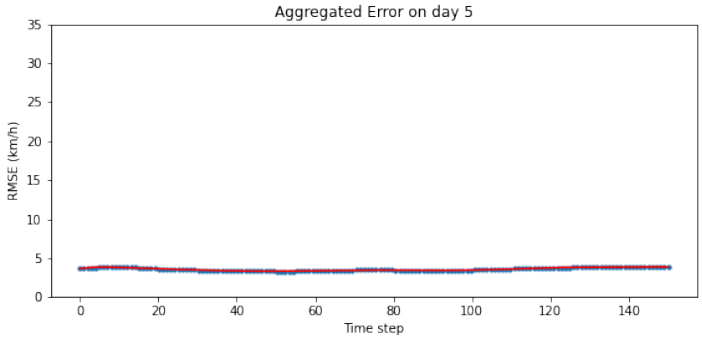
(c)


Aggregated Error on day 3
RMSE (km/h)
Time step

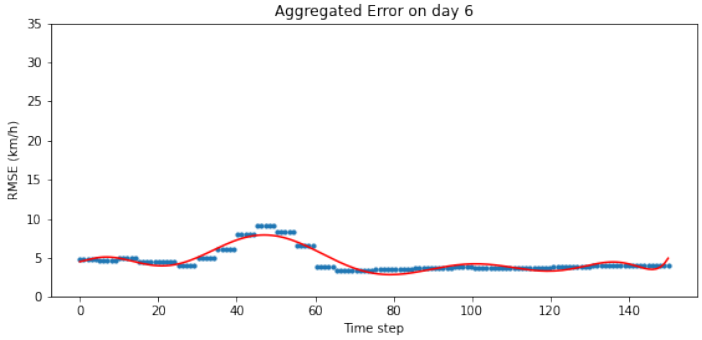
(d)


Aggregated Error on day 4
RMSE (km/h)
Time step

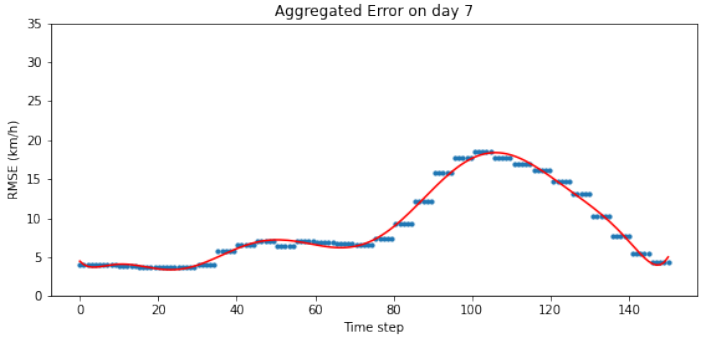
(e)


Aggregated Error on day 5
RMSE (km/h)
Time step

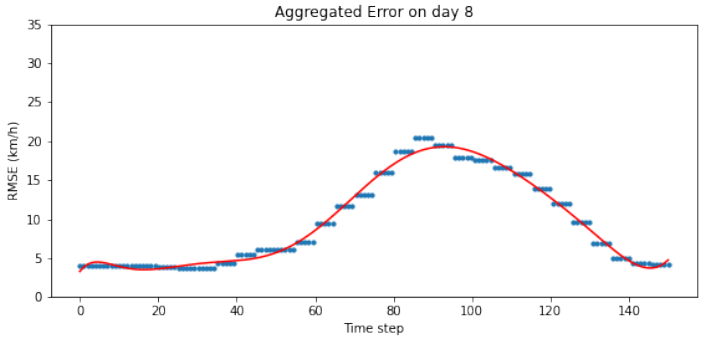
(f)


Aggregated Error on day 6
RMSE (km/h)
Time step

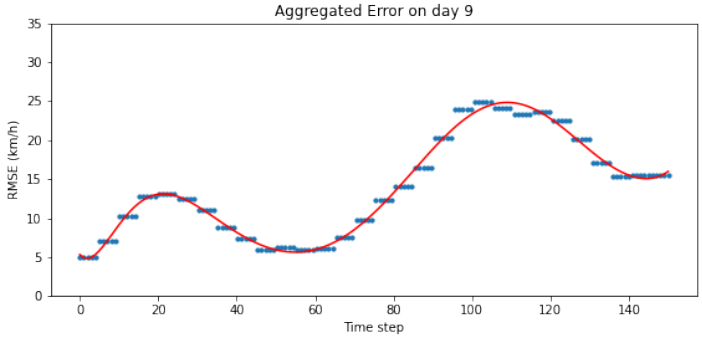
(g)


Aggregated Error on day 7
RMSE (km/h)
Time step

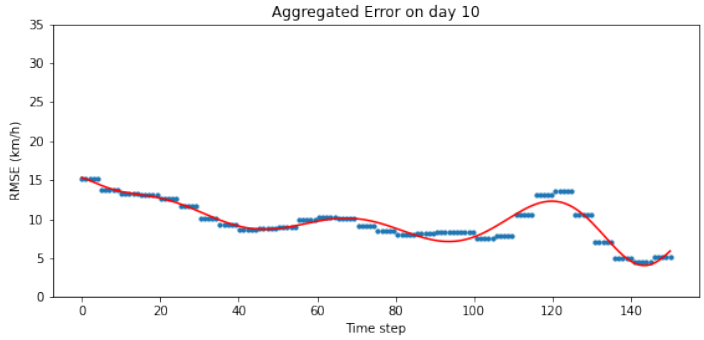
(h)


Aggregated Error on day 8
RMSE (km/h)
Time step

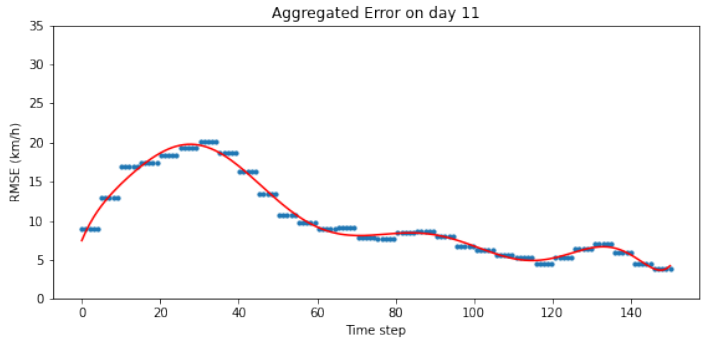
(i)


Aggregated Error on day 9
RMSE (km/h)
Time step

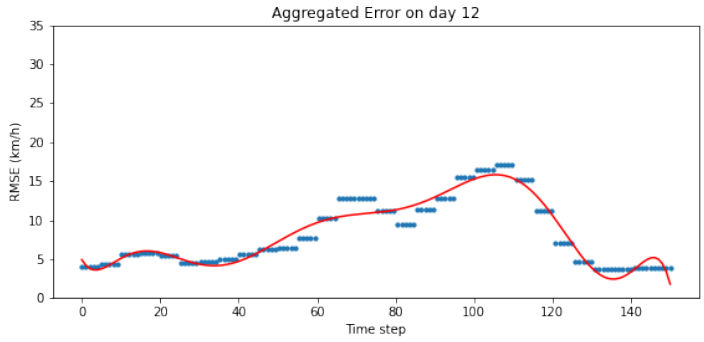
(j)



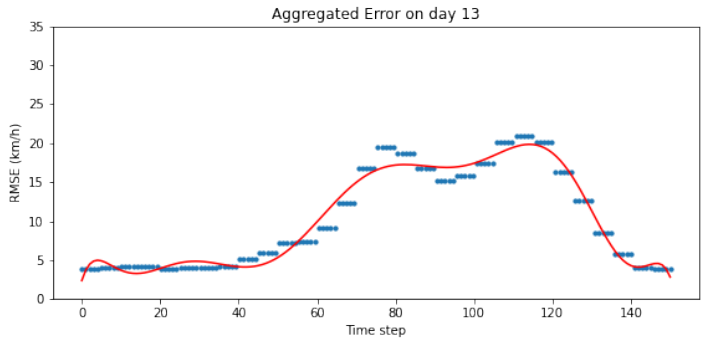
(k)



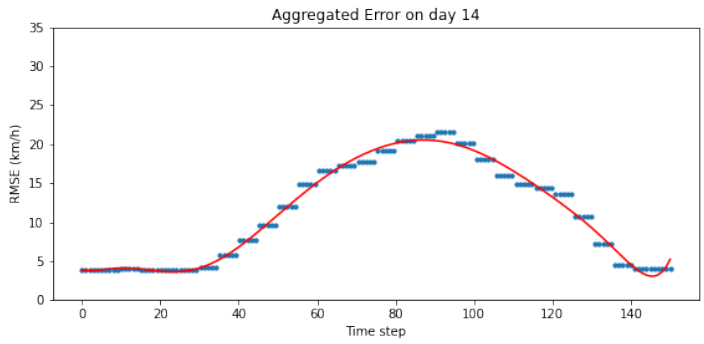
(l)



(m)

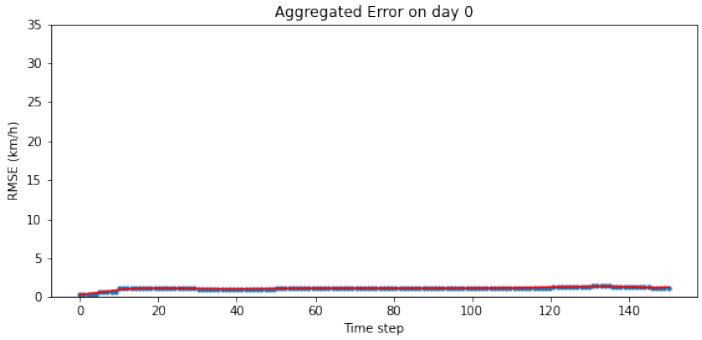


(n)

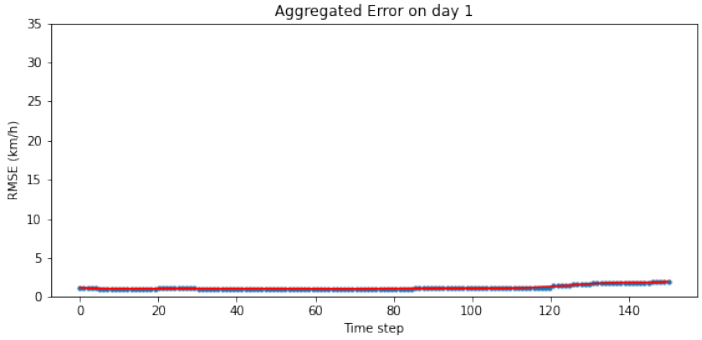


(o)

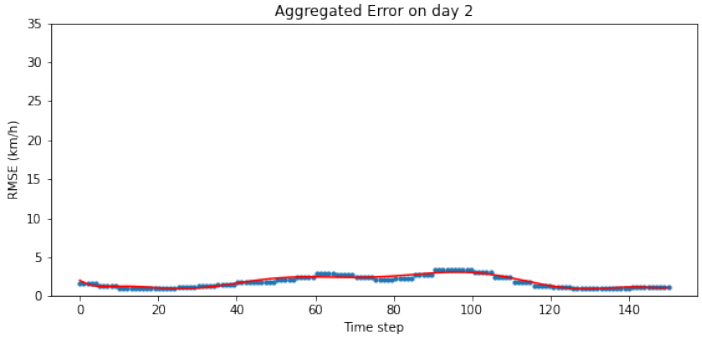
Figure A.2: Aggregated LSTM encoder-decoder prediction error distribution of 15 days


Aggregated Error on day 0
RMSE (km/h)
Time step

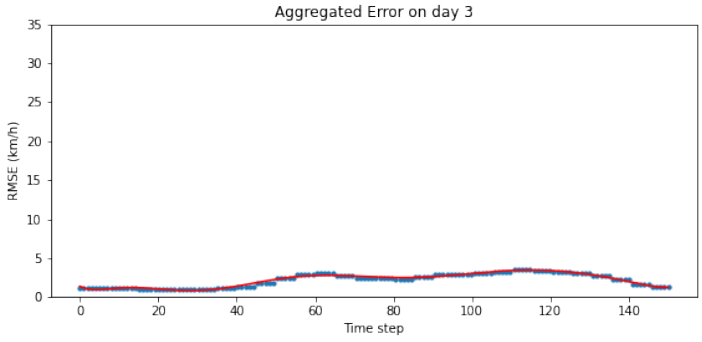
(a)


Aggregated Error on day 1
RMSE (km/h)
Time step

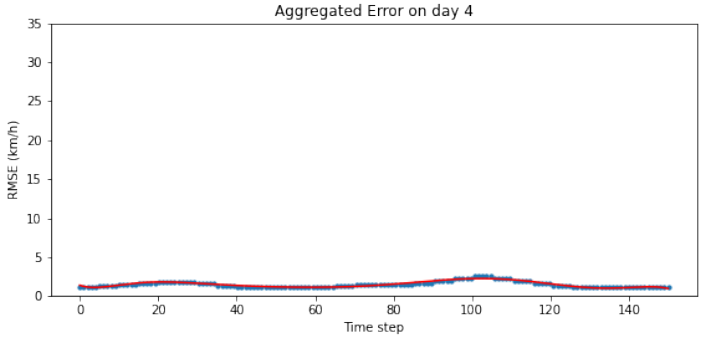
(b)


Aggregated Error on day 2
RMSE (km/h)
Time step

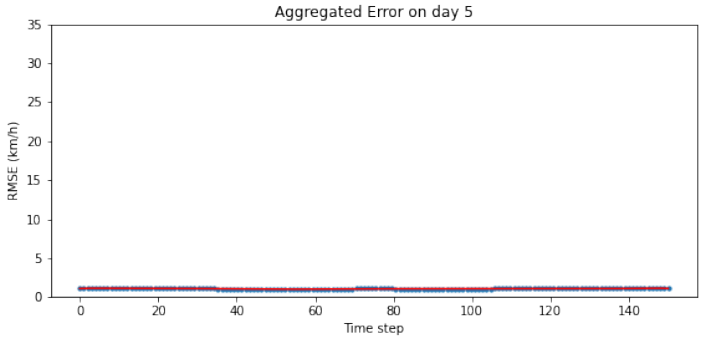
(c)


Aggregated Error on day 3
RMSE (km/h)
Time step

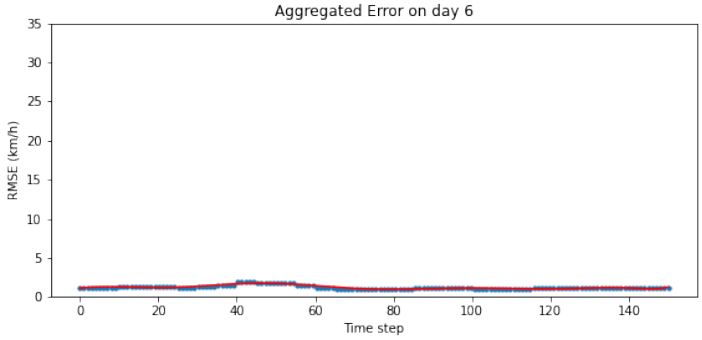
(d)


Aggregated Error on day 4
RMSE (km/h)
Time step

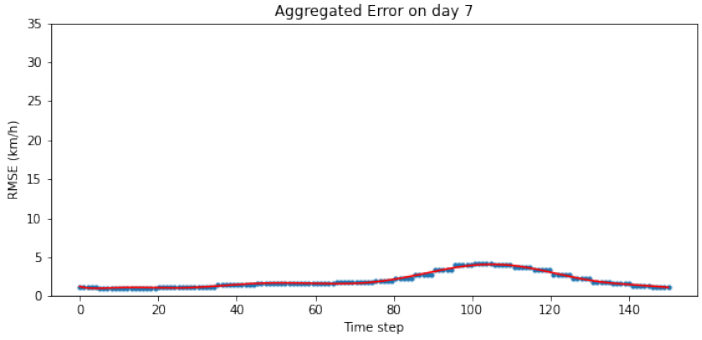
(e)


Aggregated Error on day 5
RMSE (km/h)
Time step

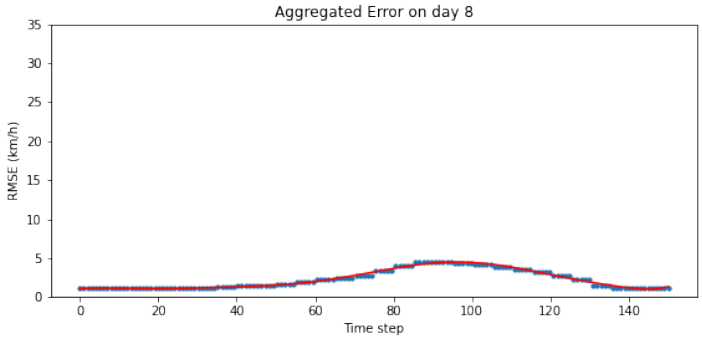
(f)


Aggregated Error on day 6
RMSE (km/h)
Time step

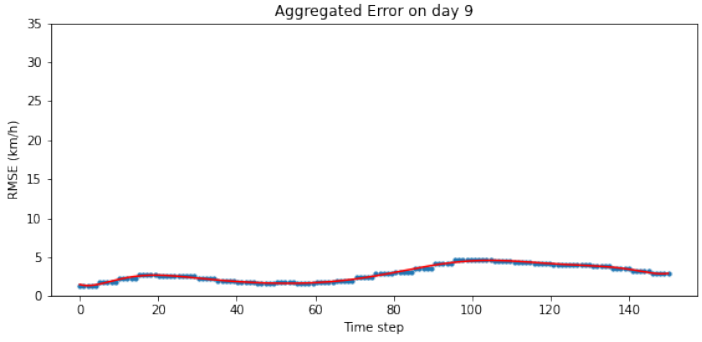
(g)


Aggregated Error on day 7
RMSE (km/h)
Time step

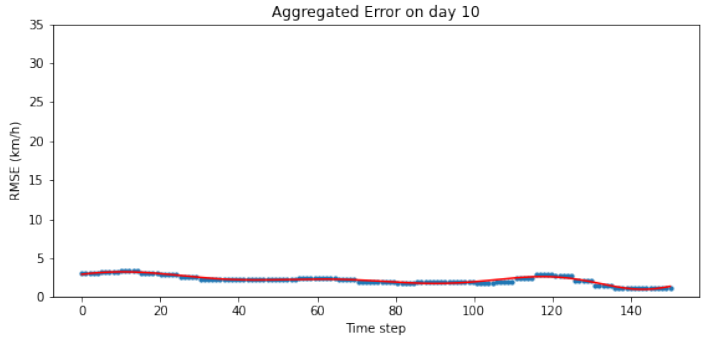
(h)


Aggregated Error on day 8
RMSE (km/h)
Time step

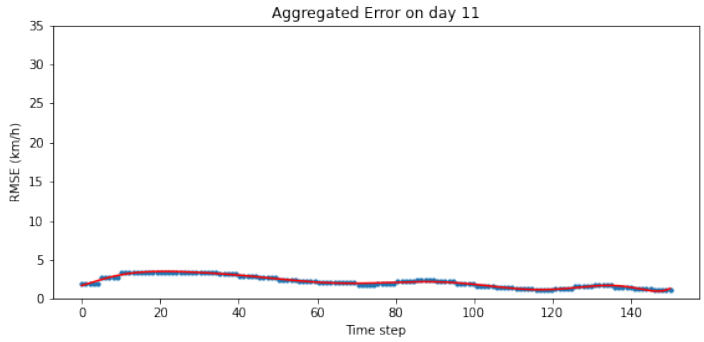
(i)


Aggregated Error on day 9
RMSE (km/h)
Time step

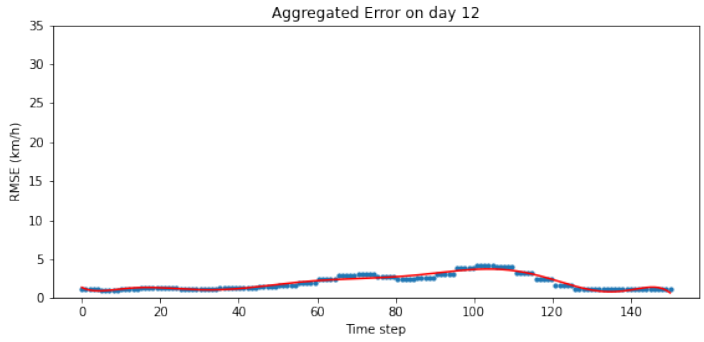
(j)



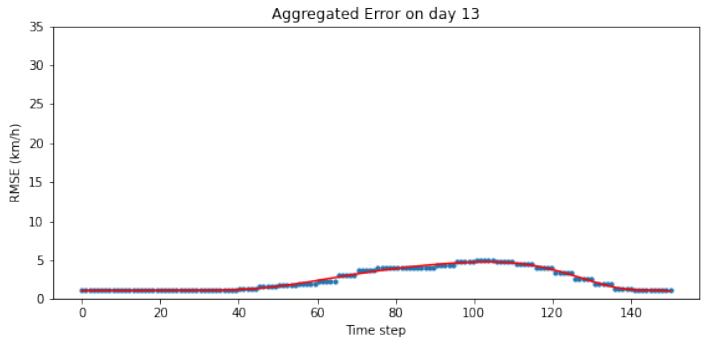
(k)



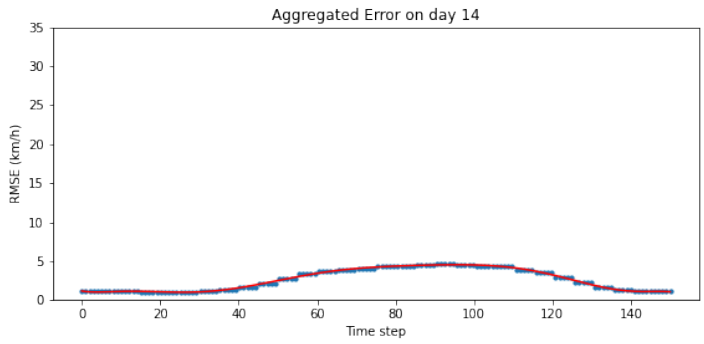
(l)



(m)

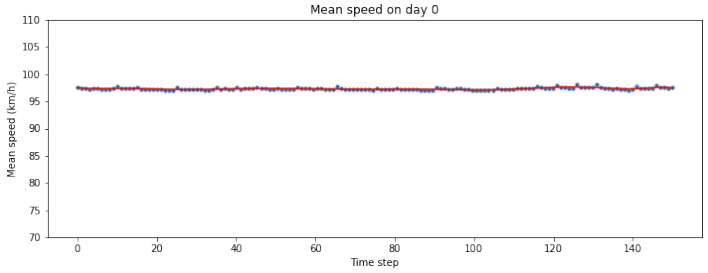


(n)

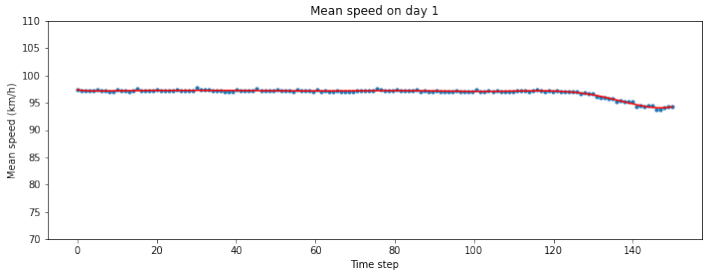


(o)

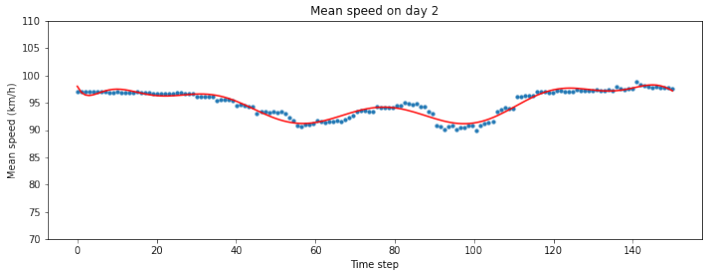
Figure A.3: Aggregated DGCN prediction error distribution of 15 days


Mean speed on day 0
Mean speed (km/h)
Time step

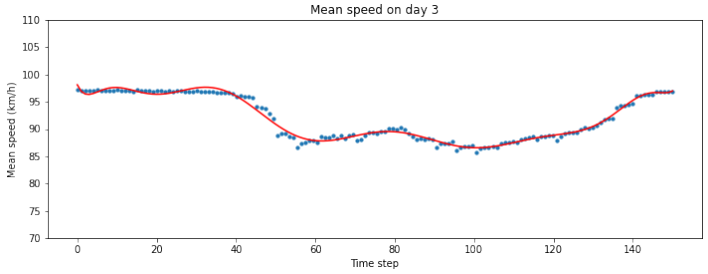
(a)


Mean speed on day 1
Mean speed (km/h)
Time step

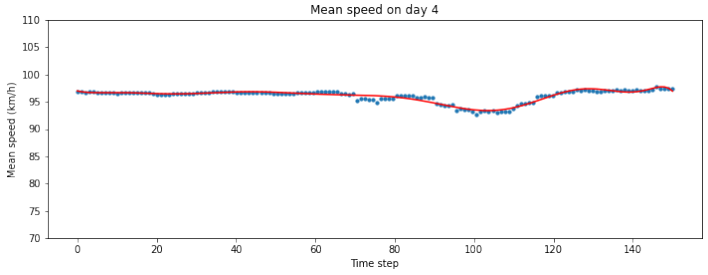
(b)


Mean speed on day 2
Mean speed (km/h)
Time step

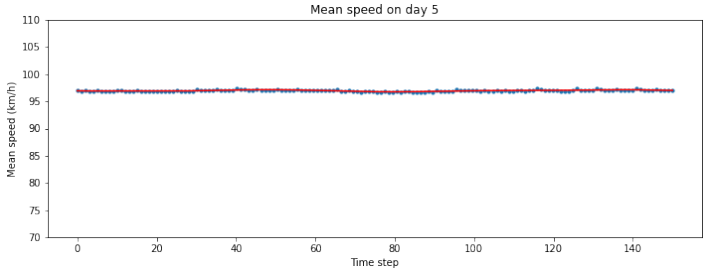
(c)


Mean speed on day 3
Mean speed (km/h)
Time step

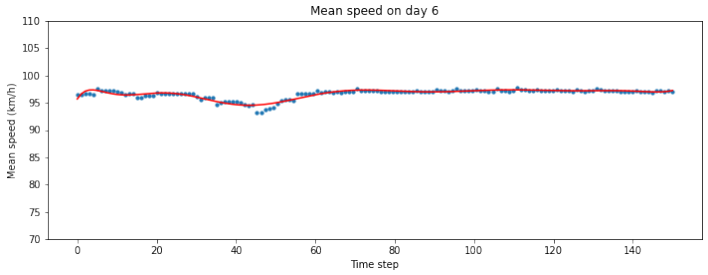
(d)


Mean speed on day 4
Mean speed (km/h)
Time step

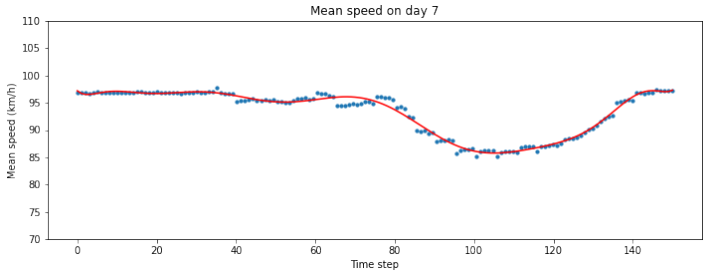
(e)


Mean speed on day 5
Mean speed (km/h)
Time step

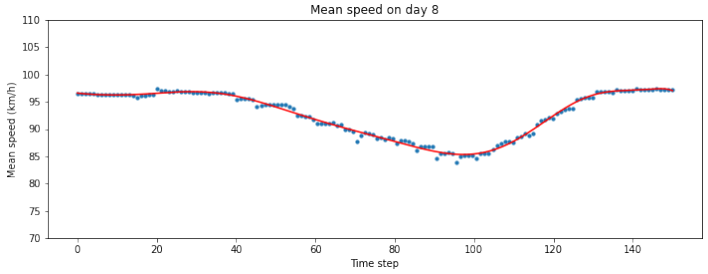
(f)


Mean speed on day 6
Mean speed (km/h)
Time step

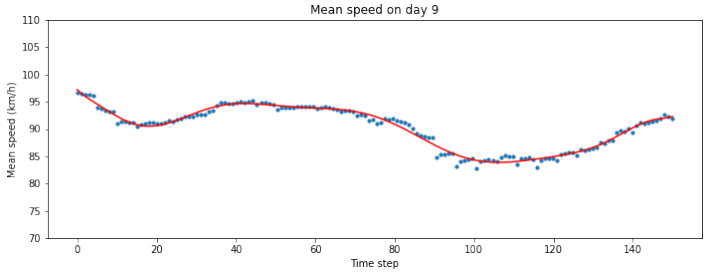
(g)


Mean speed on day 7
Mean speed (km/h)
Time step

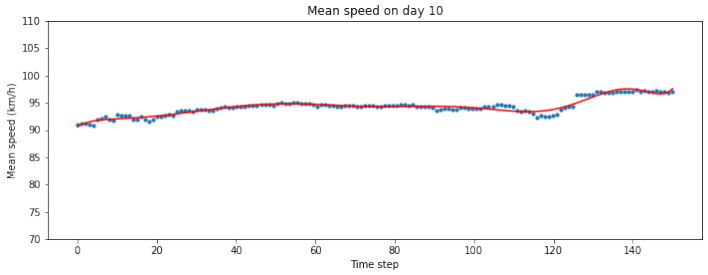
(h)


Mean speed on day 8
Mean speed (km/h)
Time step

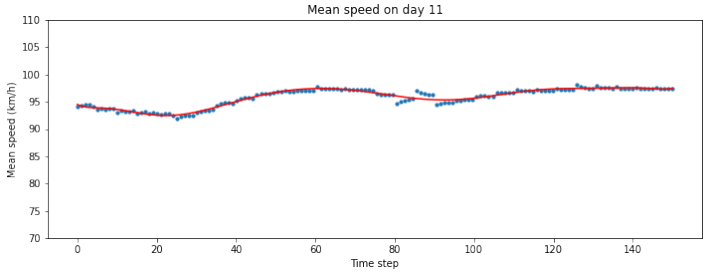
(i)


Mean speed on day 9
Mean speed (km/h)
Time step

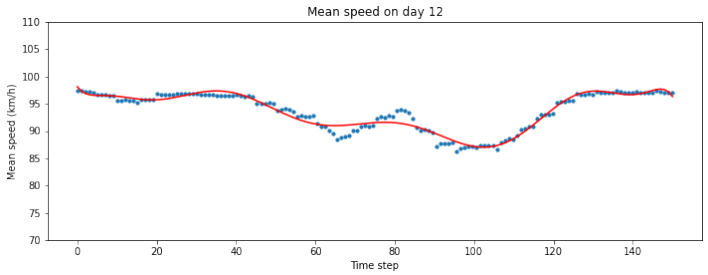
(j)



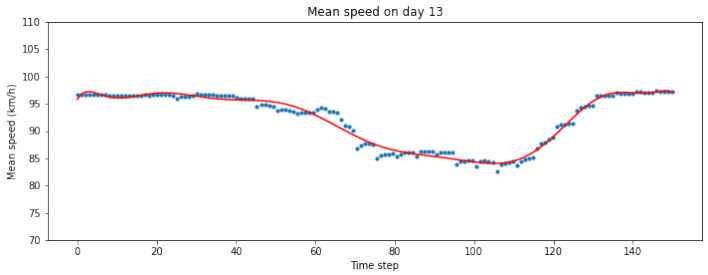
(k)



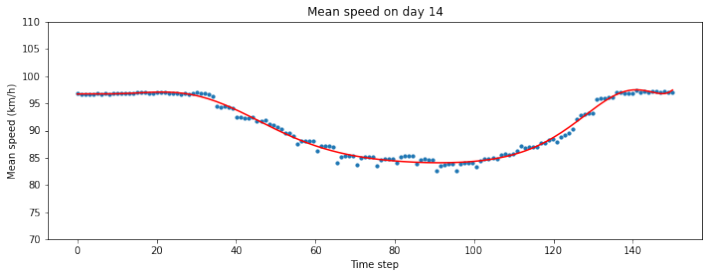
(l)



(m)

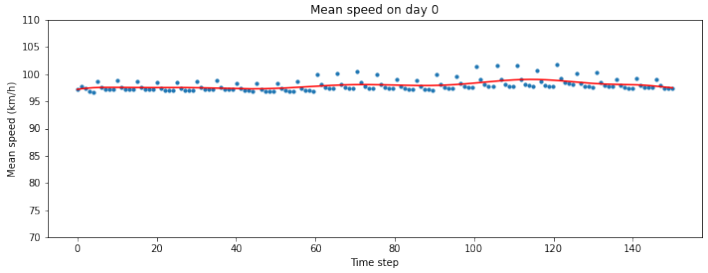


(n)

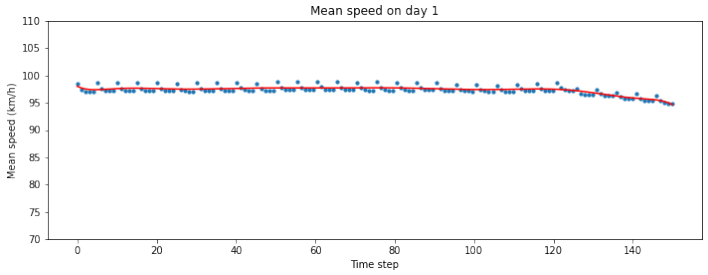


(o)

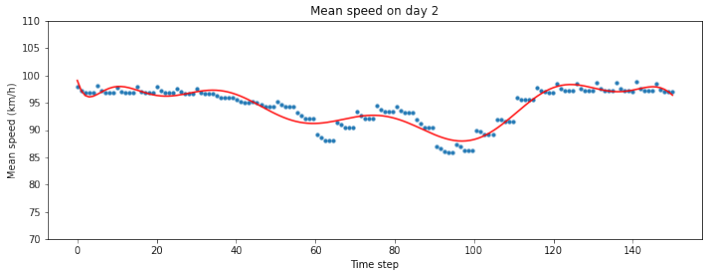
Figure A.4: Hybrid model mean speed distribution of 15 days


Mean speed on day 0
Mean speed (km/h)
Time step

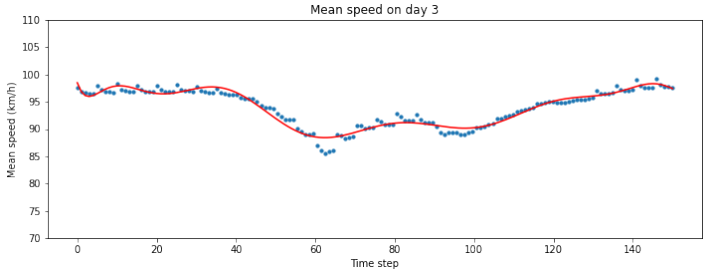
(a)


Mean speed on day 1
Mean speed (km/h)
Time step

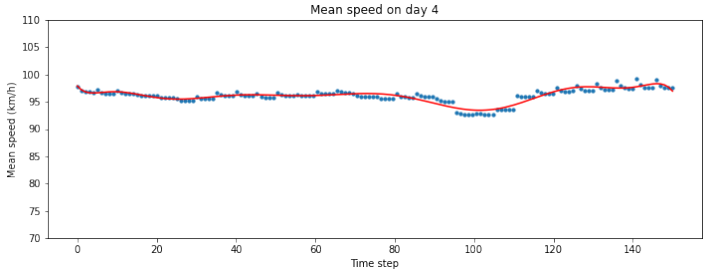
(b)


Mean speed on day 2
Mean speed (km/h)
Time step

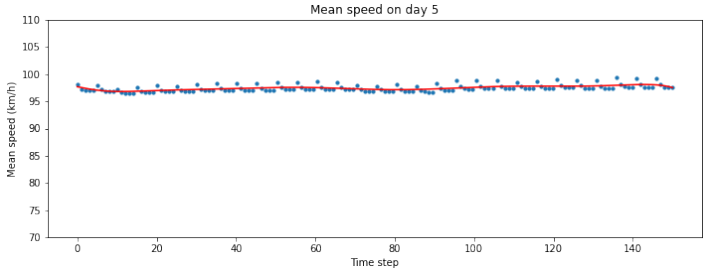
(c)


Mean speed on day 3
Mean speed (km/h)
Time step

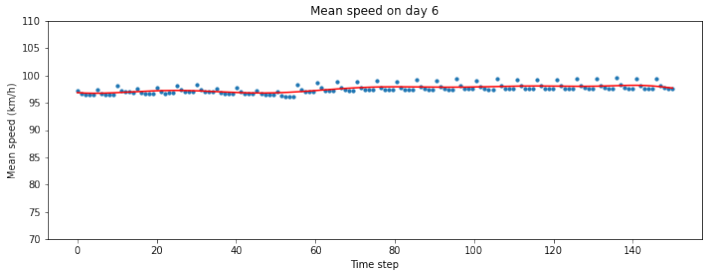
(d)


Mean speed on day 4
Mean speed (km/h)
Time step

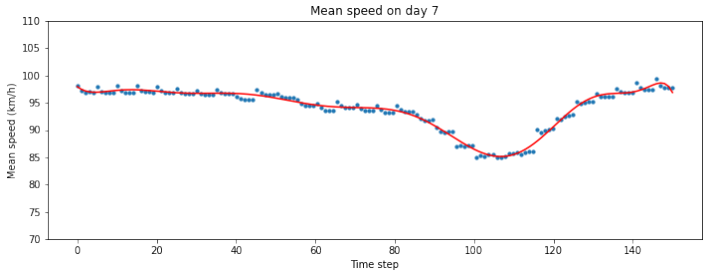
(e)


Mean speed on day 5
Mean speed (km/h)
Time step

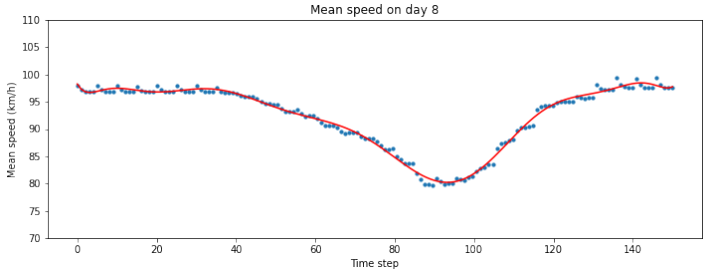
(f)


Mean speed on day 6
Mean speed (km/h)
Time step

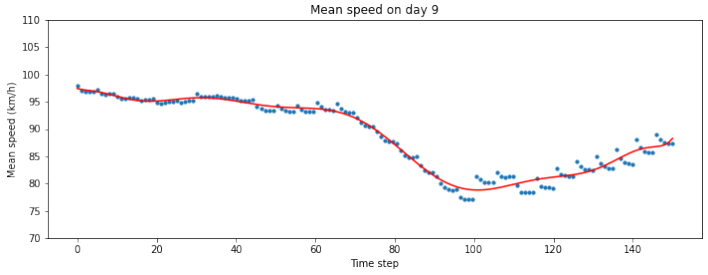
(g)


Mean speed on day 7
Mean speed (km/h)
Time step

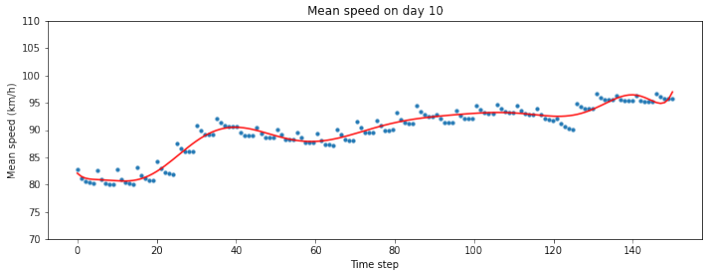
(h)


Mean speed on day 8
Mean speed (km/h)
Time step

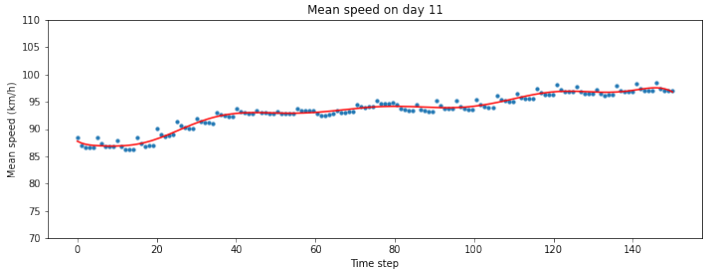
(i)


Mean speed on day 9
Mean speed (km/h)
Time step

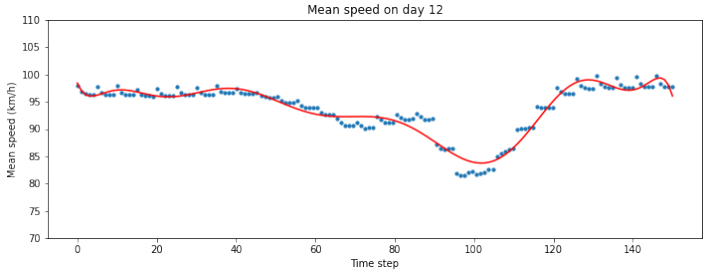
(j)



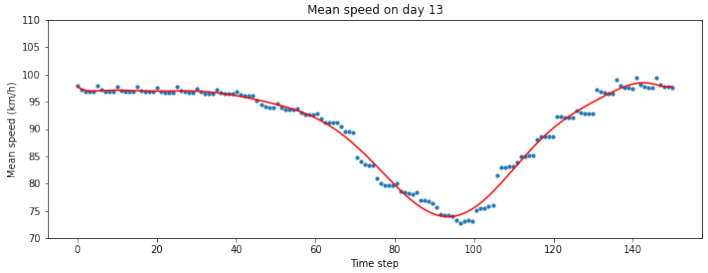
(k)



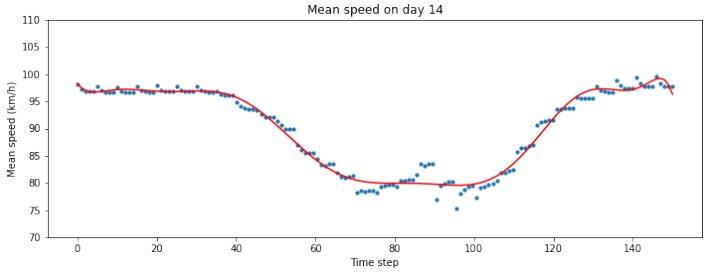
(l)



(m)

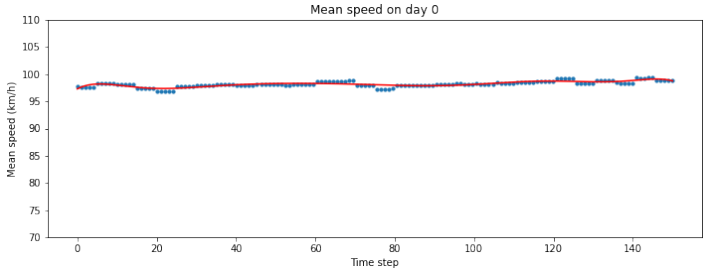


(n)

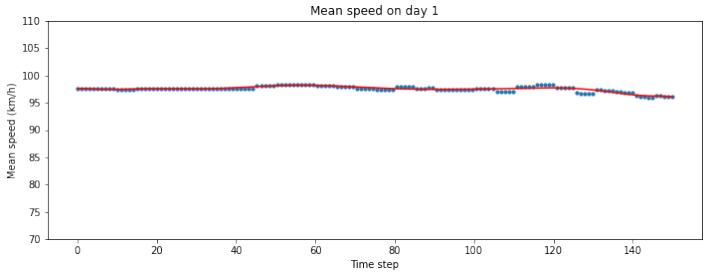


(o)

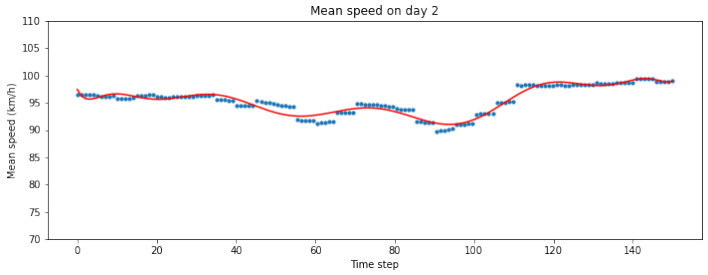
Figure A.5: LSTM encoder-decoder mean speed distribution of 15 days


Mean speed on day 0
Mean speed (km/h)
Time step

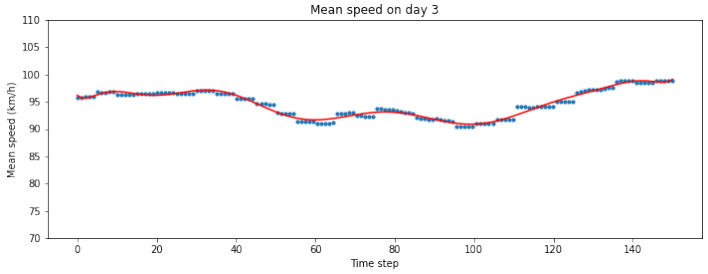
(a)


Mean speed on day 1
Mean speed (km/h)
Time step

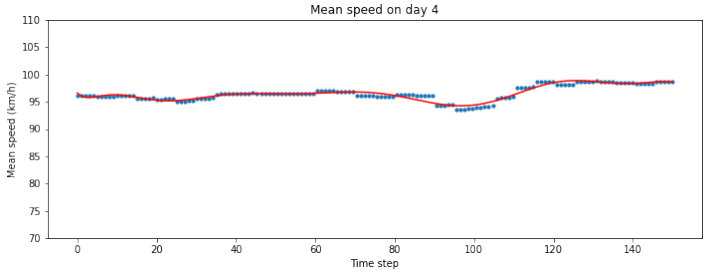
(b)


Mean speed on day 2
Mean speed (km/h)
Time step

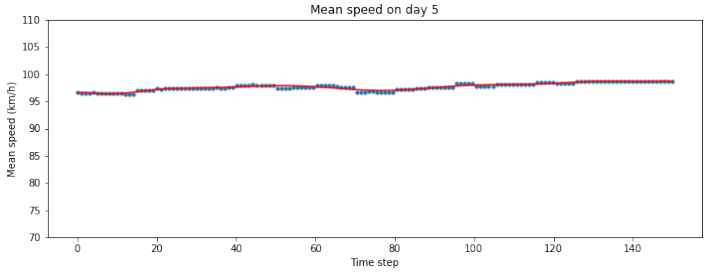
(c)


Mean speed on day 3
Mean speed (km/h)
Time step

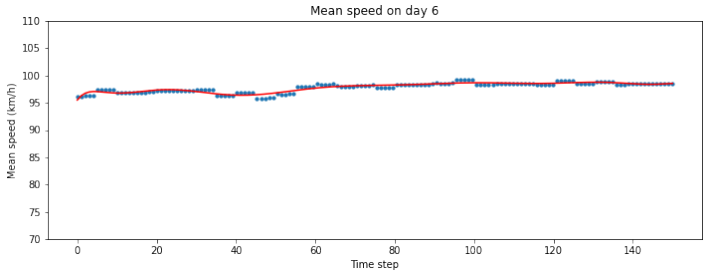
(d)


Mean speed on day 4
Mean speed (km/h)
Time step

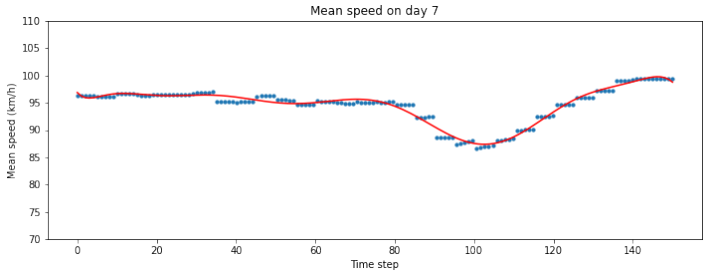
(e)


Mean speed on day 5
Mean speed (km/h)
Time step

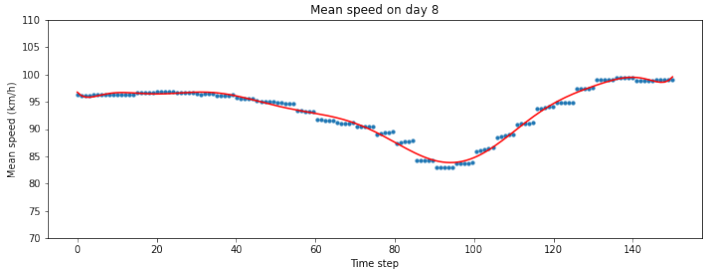
(f)


Mean speed on day 6
Mean speed (km/h)
Time step

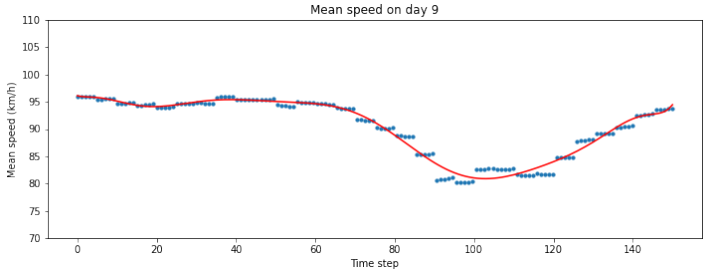
(g)


Mean speed on day 7
Mean speed (km/h)
Time step

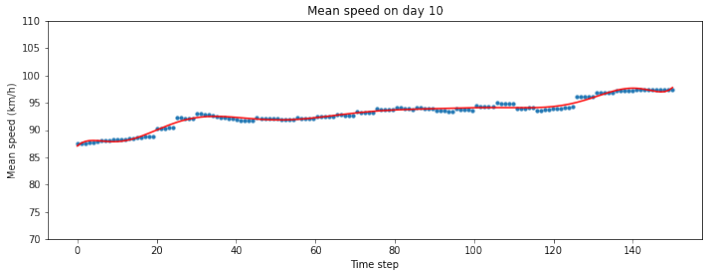
(h)


Mean speed on day 8
Mean speed (km/h)
Time step

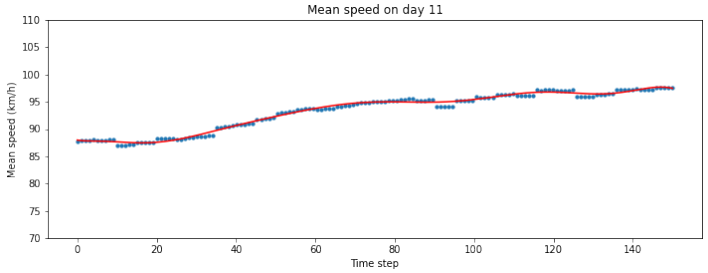
(i)


Mean speed on day 9
Mean speed (km/h)
Time step

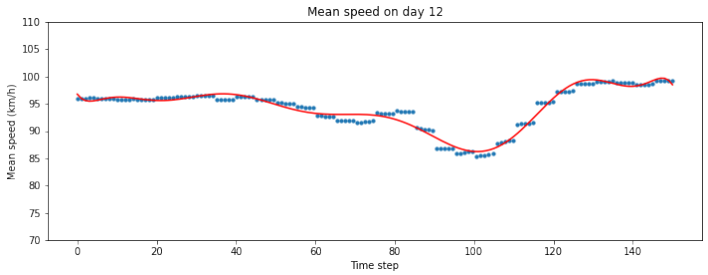
(j)



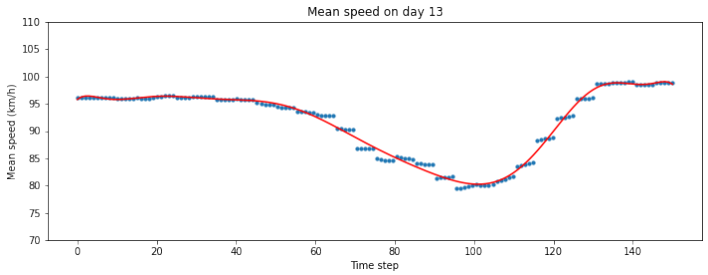
(k)



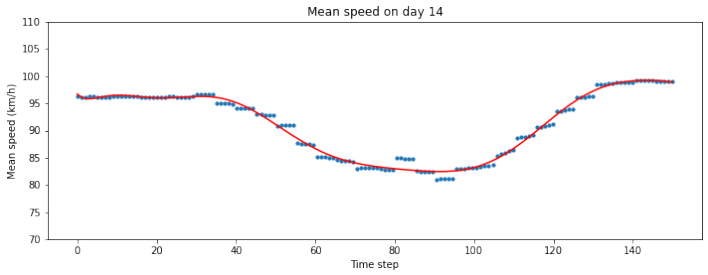
(l)



(m)



(n)



(o)

Figure A.6: DGCN mean speed distribution of 15 days