University of Sheffield

Machine Learning Methods for Autonomous Classification and Decision Making

Yifei Zhu

1st *Supervisor:* Lyudmila Mihaylova

2nd *Supervisor:* Michael Balikhin

A thesis submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

Department of Automatic Control and Systems Engineering

Declaration

Acknowledgements

Completing the PhD study over the last four years was a challenging journey. However, I would like to thank everyone who encouraged me and provided help through the journey.

First, I would like to thank my supervisor, Prof. L. Mihaylova, for providing guidance, feedback and encouragements. Second I would like to thank Dr. Peng Wang and Dr. Xingchi Liu who are co-authors of some of my publications and the journal paper under preparation, and who provided great help and guidance to the project conducted the thesis.

Secondly, I would like to express my heartfelt appreciation to my family. Thank my parents for fully supporting my further study as a PhD student. Thank my wife for making those sacrifices and accompanying me through those years.

Abstract

This thesis focuses on developing machine learning methods for autonomous classification and decision making, especially on two case studies: traffic speed prediction and cancer bone segmentation. For traffic speed prediction, the convolutional neural network (CNN) achieves state-of-the-art results in complex traffic networks. However, the pooling layers cause the loss of information within the data. This thesis proposes an efficient capsule network for traffic speed prediction. The proposed capsule network replaces the pooling layer with capsules connected by dynamic routing and encodes the features and probability of those features showing on the local region. The proposed capsule network provides outperformed results compared to state-of-the-art CNNs. However, the CNN and capsule network (CapsNet) are parametric models and the uncertainty is, thus, not analysed. Two Gaussian process (GP) frameworks are proposed for traffic speed prediction, equipping the CNN with the ability to quantify uncertainty. The first framework proposes to equate a state-of-the-art CNN with a shallow GP. The proposed approach is evaluated and the uncertainty is analysed by applying the confidence interval. In addition, the impact of the noise is investigated by adding a different level of noise. The second framework is a novel deep kernel CNN-GP framework with spatio-temporal kernels, allowing it to abstract high-level features and consider both time and space. The proposed CNN-GP framework is validated and evaluated using CO_2 concentration and traffic prediction for the short-term and long-term. An efficient uniform error bound is proposed and evaluated with simulated and real data. For cancer bone segmentation, machine learning methods are proposed to segment bone lesions in cancer-induced bone disease from Micro Computed Tomography (µCT) images, which brings a new perspective of dealing with bone caner segmentation. The performances are evaluated and their effectiveness is compared. Due to the limited number of datasets and the lack of labelled lesions within the dataset, an approach to generate simulated data is proposed. With an enhanced dataset, a generative adversarial network is proposed to reconstruct the bone with a lesion to a healthy bone. Consequently, the location of the lesion can be obtained by subtracting the original image from the reconstructed image.

Contents

List of Symbols

Γ	Adjustable error term corresponding to a weighted least squares cost function
$\mathbb{C}$	Covariance Function
$\mathbb{E}$	Mean Function
$\alpha(\cdot)$	Layer output
θ	Model parameters
$\mathbf{A_{i,j}}$	Layer Outputs of a Network
$\mathbf{b}$	Bias
$\mathbf{Cell}(\cdot)$	Cell block
$\mathbf{CO}$	CO_2 Concentration
$\mathbf{C}_{i,j}$	Coupling coefficient
$\mathbf{D}$	Discriminator
$\mathbf{f}(\cdot)$	Forget gate
$\mathbf{G}$	Generator
$\mathbf{i}(\cdot)$	Input gate
$\mathbf{I}$	Identical Matrix
$\mathbf{K}(\cdot,\cdot)$	Kernel Function

$\mathbf{M}(\cdot)$	Memory block
$\mathbf{n}_t$	White Noise
$\mathbf{o}(\cdot)$	Output gate
$\mathbf{squash}_j$	Squashing function
$\mathbf{sum}$	Weighted sum
$\mathbf{S}$	Traffic Speed
$\mathbf{V}$	Traffic Volume
$\mathbf{W}$	Weights
$\mathbf{y}$	Ground Truth/Actual Value
$\mathbf{z}$	Intermediate Value/Feature Extracted by CNN
$\mathcal{E}_S$	Spatial edge
$\mathcal{E}_T$	Temporal edge
$\mathcal{L}(\cdot)$	Loss function
μ	Mean
$\omega_{\sigma N}(\cdot)$	Modulus of continuity
$\Phi(\cdot)$	Auto regressive operator
$\phi(\cdot)$	Nonlinear Activation
σ^2	Variance
σ_n^2	Variance of Noise
$\mathbf{flatten}(\cdot)$	Flatten operation
$\mathbf{pool}(\cdot)$	Pooling operation
$\mathbf{Prob}(\cdot)$	Probability

Θ	Moving average operator
$\widehat{\mathbf{y}}$	Model Predictions
C_l	Network Channel
$h(\cdot,\cdot)$	Binary split function
I	Information gain
L	Marginal likelihood
L_f	Lipschitz constant of unknown function
$L_k^{\partial i}$	Lipschitz constant of partial derivative kernels
$m(\cdot)$	Mean function
$max(\cdot)$	Maximum
$N(\cdot)$	Number of points derived from the discretisation of a segment
p_r, p_t	Discrete points within road segments

List of Abbreviations

AI	Artificial Intelligence
AMDS	Asymmetrical Multidimensional Scaling
ANN	Artificial Neural Network
ARIMA	Auto Regressive Integrated Moving Average
AST	Adaptive Spatio-temporal
AU	Aleatoric Uncertainty
BCNN	Bayesian Combined Neural Network
CapsNet	Capsule Network
CI	Confidence Interval
CNN	Convolutional Neural Network
Convnet GP	Deep Convolutional Neural Network as Shallow Gaussian Process
DBN	Deep Belief Network
DCGAN	Deep Convolutional Generative Adversarial Network
DCGAN	Deep Kernel Learning
DKL	Deep Kernel Learning

DL	Deep Learning
ELBO	Evidence Lower Bound
ELU	Exponential Linear Unit
EU	Epistemic Uncertainty
FCN	Fully Convolutional Network
FP	False Positive
FN	False Negative
GAN	Generative Adversarial Network
GP	Gaussian Process
HDNN	Hybrid Deep Neural Network
IOU	Intersection of Union
ITS	Intelligent Transportation Systems
KL	Kullback-Leiber
LSTM	Long Short-Term Memory
LS-SVM	Least Square Support Vector Machine
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
ML	Machine Learning
MS	Multi-scale Version
MSE	Mean Square Error
μCT	Micro Computed Tomography

NN	Neural Network
NRMSE	Normalized Root Mean Square Error
PCA	Principal Component Analysis
RBF	Radial Basic Function
RBM	Restrict Boltzmann Machine
ReLu	Rectified Linear Unite
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SAE	Stacked Autoencoder
SE	Structured Edge
SH	Sharpening
SRK	Segment-based Regression Kriging
SSNN	State Space Neural Network
SSIM	Structural Similarity
STGP	Spatio-temporal Gaussian Process
SVM	Support Vector Machine
TDNN	Time-delay Neural Network
TP	True Positive
UEB	Uniform Error Bound
UQ	Uncertainty Quantification
VI	Variational Inference

List of Figures

List of Tables

Chapter 1

Introduction

Data has been the lifeblood of a great number of domains, and the rapid growth of data leads to the increasing demand for intelligent automation. Machine learning (ML) is a powerful tool for autonomous data analysis and decision making in different domains. Before ML, data is analysed by model-based methods or by humans, which are either expensive or time consuming. For example, different physical models are required to be designed under different traffic scenarios and computerised tomography (CT) images for patients can only be analysis by experienced doctors. ML shows the ability to handle large amounts of data and decision making, which has revolutionised the field of autonomous systems. ML is essentially a data-driven methods are developed to investigate the inherent relationships in historical data. A wide range of approaches, including decision trees [7], K-nearest neighbor [8], support vector machines [9], statistical methods and ending with artificial neural networks (ANNs) [10], are the categories of ML. Deep learning (DL) methods are one of the subsets of ML, and have achieved great success, especially convolutional neural networks (CNNs) [11]. However, DL methods only provide deterministic predictions, but the degree of trust of the results is not analysed and not well studied.

The focus of this thesis is the development of novel ML methods for autonomous classification and decision making. Classification is one of the most fundamental problems in machine learning, which involves assigning data into different predefined categories based on their properties. The decision-making problem involves choosing the most appropriate action based on the current state, and regression is one representative. Both problems are similar, since discrete predictions are made for labels in the classification problem, and continuous predictions are made for the regression problem. In this thesis, two case studies are investigated: 1. traffic speed prediction in a complex traffic network; 2. cancer bone and lesion segmentation. For both case studies, the inputs are essentially images, since the time series of traffic speed is converted into an image, and CT image slides are images. In this thesis, the ML methods developed focus on analysing image data.

1.1 Objective

This thesis aims to present ML methods for autonomous classification and decision making, and implement them in areas that are challenging and have not been previously researched. The impact of uncertainty is investigated. Two case studies are applied, as both focus on image data.

1.2 Thesis Outline

This thesis consists of five chapters, and the structure is outlined as following:

Chapter 1 introduces the topic and objectives of this thesis. The outline and key contributions in each chapter are emphasised. Relevant publications are listed in the last section of this chapter.

Chapter 2 reviews existing machine learning methods for regression and classification, followed by an overview of uncertainty quantification. The main existing machine learning methods for traffic prediction are introduced, and the convolutional neural network is emphasised as the baseline for this thesis. The types of uncertainty and uncertainty quantification methods are demonstrated. A review of fast edge detection using structured forests for classification and segmentation is provided.

Chapter 3 focuses on machine learning methods for traffic speed prediction and uncertainty quantification. In this case study, the aim is to present machine learning methods that accurately predict traffic speed in the future and quantify the uncertainty. The traffic speed data is converted into an image as proposed in the literature, and an efficient Capsule Network (CapsNet) for complex traffic networks is proposed to overcome the drawbacks of the state-of-the-art Convolutional Neural Network (CNN) for traffic prediction. Since CNN and CapsNet are parametric models and provide deterministic results, the uncertainty is not analysed. Therefore, two Gaussian process (GP) related frameworks, deep CNN as shallow GP (ConvNet GP) and CNN with GP regression (CNN-GP), are proposed for traffic speed prediction, which equip the CNN with the ability of uncertainty quantification. The uncertainty is analysed by applying the confidence interval. In addition to the confidence interval, a uniform error bound is modified and applied for the CNN-GP framework to quantify the uncertainty. The proposed approaches are evaluated with real traffic speed data, and the impact of noise is investigated by adding different levels of noise to the data. Finally, a spatio-temporal kernel is proposed based on the CNN-GP framework, since the traffic speed data show periodicity, which provides further improvement in the prediction accuracy.

Chapter 4 focuses on machine learning methods for cancer bone segmentation. In this case study, the aim is to locate the position of the lesions and quantify the areas of the lesion. Four machine learning approaches are proposed for bone segmentation, including a CNN, a CapsNet, a Fully Convoltuional Network (FCN) and a ConvNet GP, which brings a new perspective of dealing with bone cancer segmentation. The performances are evaluated and the effectiveness is compared. Because the number of the dataset is limited and the lesions in the dataset are not labelled, an approach to generate a simulated data set is proposed. Furthermore, the major challenge of this case study is that the lesion area is identical as the background. Therefore, instead of directly segmenting the lesion, a generative adaptive network (GAN) is proposed to reconstruct the bone with the lesion back to healthy, and thus the location of the lesion can be obtained by subtracting the original image and the reconstruction image. The performance is evaluated and shows great potential towards fully automatic bone tumour segmentation.

Chapter 5 gives a summary of the thesis and discussions of future work.

1.3 Key Contributions

Chapter 3: This chapter delves into ML methodologies tailored for time series prediction, paired with uncertainty quantification. The groundbreaking contributions encapsulated within this chapter include:

- **Image Conversion of Time Series Data**: Distinct from traditional numerical representations of time series data, this chapter adopts a pioneering approach, as previously illuminated in select literature, where in traffic speed data undergo transformation into a visual image format, enabling the application of image-focused algorithms.
- **Novel CapsNet for Time Series Prediction**: Recognising inherent limitations within the current CNN paradigm, this chapter proposes an efficient CapsNet. This introduces concepts of capsule and dynamic routing algorithm, aiming to overcome identified drawbacks of CNNs in the domain of time series forecasting [C1].
- **Incorporation of Uncertainty Quantification Frameworks**: A salient

gap observed in the deterministic outputs of CNN and CapsNet is the absence of uncertainty quantification. Addressing this, this chapter pioneers the introduction of ConvNet GP [C3]. These architectures endow CNN with the indispensable faculty of quantifying inherent uncertainties.

- **Deep Learning Kernel**: A novel CNN-GP approach is proposed for prediction with time series data. The advantages of GP regression are augmented with deep learning kernels, and this provides both efficient feature extraction, robustness to uncertainties, and accurate results [J1].

- **Spatio-temporal kernel**: Acknowledging the periodicity intrinsic to traffic speed data, this chapter heralds the creation of a spatio-temporal kernel, anchored upon the CNN-GP approach. This kernel embodies a refined understanding of temporal patterns, significantly improving the accuracy of short-term prediction, and enabling long-term prediction [J1].

- **Empirical Noise Impact Analysis**: The uncertainties are investigated through the confidence interval. Beyond the baseline evaluation using authentic traffic speed data, the research delves deep into the models' robustness, studying their behaviour and efficacy in the face of varied noise magnitudes introduced in the dataset.

- **Advanced Uncertainty Analytical**: An efficient adaptation of the uniform error bound has been specifically devised and integrated within the CNN-GP framework, offering a cutting-edge technique for uncertainty quantification.

Chapter 4: In this chapter, the focus is on innovative machine learning methodologies designed for cancer bone segmentation. The salient contributions and groundbreaking methodologies delineated in this chapter are as follows:

- **Multifaceted Machine Learning Architectures**: this chapter introduces four distinct ML paradigms for the intricate task of bone segmentation: traditional dense CNN, CapsNet, FCN and ConvNet GP. The inclusion of these diverse methodologies underscores a novel and comprehensive perspective toward the exigencies of bone segmentation [C2].

- **Simulated Dataset Generation**: Given the inherent constraints of the dataset, its limited volume, and the absence of lesion labelling, this chapter innovatively proposes an approach for the generation of simulated datasets. This approach not only augments the existing dataset but also addresses the gap in terms of labelling instances.
- **Challenging Lesion-Background Indistinguishability**: A prominent obstacle identified in this domain is the uncanny resemblance between the lesion area and the background. Instead of conventionally attempting direct lesion segmentation, this work breaks new ground by suggesting the deployment of a GAN. The GAN's principal role is to reconstruct bones inflicted with lesions back to their pristine, lesion-free state. The precise location of the lesion is subsequently discerned by juxtaposing the original and reconstructed images. Although unconventional, this approach has been shown to have significant potential.

1.4 Publications

The author's publications with relevance to this thesis are listed as following:

Peer Reviewed Papers in Conference Proceedings

[C1] Y. Kim, P. Wang, Y. Zhu and L. Mihaylova, "A Capsule Network for Traffic Speed Prediction in Complex Road Networks" in *Proc. of 2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2018, pp. 1-6, doi: 10.1109/SDF.2018.8547068.

[C2] Y. Zhu, A. C. Green, L. Guo, H. R. Evans and L. Mihaylova, "Machine Learning Approaches for Cancer Bone Segmentation from Micro Computed Tomography Images,"In *Proc. of 2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020, pp. 1-6, doi: 10.23919/FUSION45008.2020.9190495.

[C3] Y. Zhu, P. Wang and L. Mihaylova, "A Convolutional Neural Network Combined with a Gaussian Process for Speed Prediction in Traffic Networks," In *Proc. of 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021, pp. 1-7, doi: 10.1109/MFI52462.2021.9591204, **Best student paper award**.

Peer Reviewed Journal Paper

[J1] Y. Zhu, X. Liu, R. Lane, N. Bouaynaya and L. Mihaylova, "A Convlutional Neural Network-Gaussian Process Approach with Deep Kernels for Time Series Prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2023, under review.

Chapter 2

Literature Review

2.1 Backgrounds

While a number of definitions of artificial intelligence (AI) have been proposed over the last decades, John McCarthy stated that artificial intelligence is the science and engineering of making intelligent machines and computer programs. AI is related to the similar task of using computer to understand human intelligence [12]. ML is an application of AI that builds a model based on historical data to make decisions or predictions without the need of explicit programming. Tom Mitchell proposed that "Machine learning is a computer program to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [13]."

Generally, there are three types of machine learning: supervised learning methods [14], unsupervised learning [14, 15] and reinforcement learning [14, 16]. For supervised learning, an input data set and the correct outputs are given. Supervised learning problems are classified into regression and classification problems. In regression problems, the target is to predict the results within a continuous domain, which means that the input data are mapped to some continuous functions. In classification problems, the targets are instead to predict the results in a discrete domain, which means the input data are mapped into discrete categories. For example, historical data of the stock market are given to predict the stock price in the next fifteen minutes. For unsupervised learning, limited or even no knowledge of the results are provided. Unsupervised learning is a machine learning technique that allows the model to discover patterns and information on its own. It mainly deals with unlabelled data. For example, unsupervised learning is applied on a shopping website, such as Amazon, to recommend items that customers may like. For reinforcement learning, it is inspired by behaviorist psychology. The software agent learns to achieve a goal in an environment. In reinforcement learning, the model faces a game-like situation that requires the model to provide a sequence of decisions. During the training session, the model employs trail and error to come up with a solution to the problem in a complex environment. To get the model to do what it is expected to do, rewards and penalties of the actions that the model performs are applied. The goal of the model is to maximise the total rewards and minimise the total penalties. For example, AlphaGo Zero [17] was trained to defeat a world champion in the ancient Chinese game by using reinforcement learning. Machine learning has been widely used in human life, and artificial neural network (ANN) has become one of the most important algorithms of machine learning.

An outstanding advantage of ANNs is that they are data-driven models and, therefore, more accurate from physical systems with large inputs [2]. An ANN is a machine learning algorithm inspired by the human brain, as it consists of an interconnected network of artificial neurons that learns from the data by modifying its connection [10]. Like the human brain, ANNs have been widely used in massive domains, such as speech recognition, data recovery, and word recognition. More specifically, an ANN can be designed for data classification and regression. An artificial neuron, the most basic structure of an ANN, is a mathematical function that receives one or more inputs to produce an output. Fig. 2.1 shows an example of the architecture of an artificial neuron.

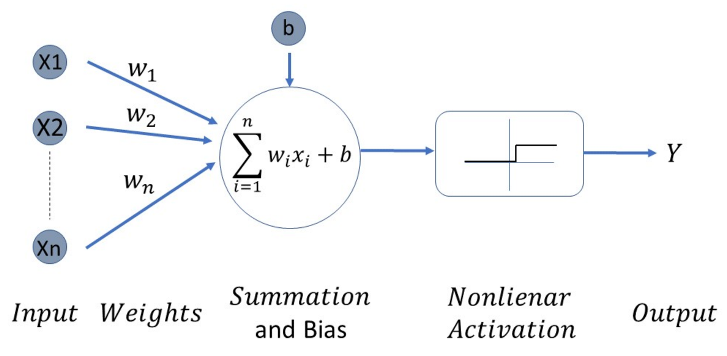


Figure 2.1: *Artificial neuron architecture.*

For a given artificial neuron, let there be n inputs labelled from X_1 to X_n. The general transformation of the artificial neuron is defined as follows:

$$a_n^l = \phi(\sum_{i=1}^{n} w_i x_i + b), \tag{2.1}$$

where w_i is weight for each input, b is the bias and ϕ represents the nonlinear activation.

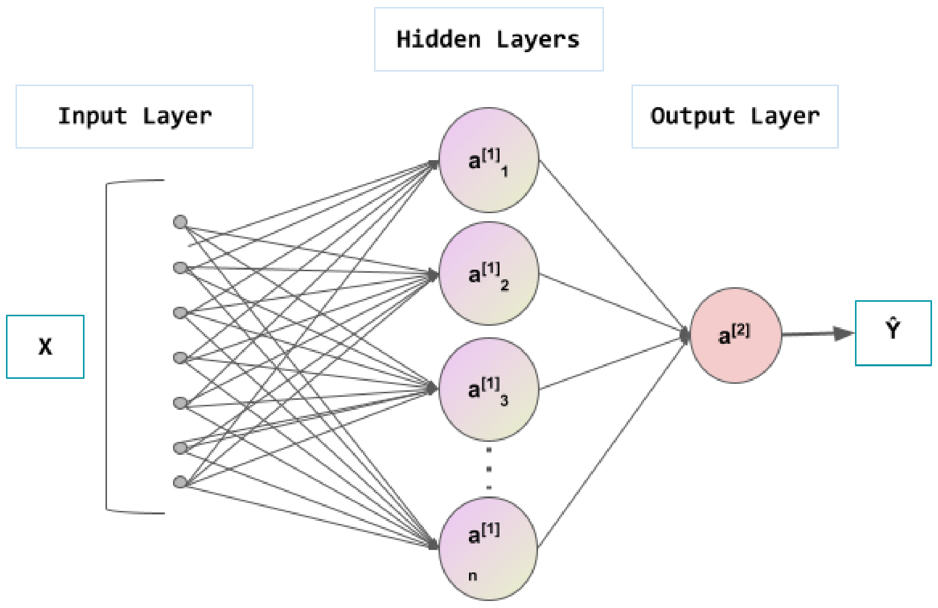


Figure 2.2: *Artificial neuron architecture.*

However, a neuron cannot work alone as the human brain is a network of neurons that cooperate toward a target. An arbitrary number of artificial neurons consist of layers in the ANNs. By simulating biological neurons, an artificial neuron transfers its input to the others to establish an interactive network. Fig. 2.2 represents an example of a 3-layer ANN architecture that contains three basic types of layers: input, hidden and output layer. The neurons in the input layer receive data. The hidden layer contains an arbitrary number of artificial neurons interconnected to neurons in the input layer. In most networks, each neuron in the hidden layer receives outputs from all neurons in the previous layer [18]. In the last layer, the output layer, the neuron receives the output from artificial neurons in the hidden layer and generates the output of the network. As the structure of ANNs becomes more complex, the term deep learning is defined to refer to ANNs with complex multilayers [19]. The difference between DL and ANNs is that the architectures of DL are more complex and the number of data is larger compared with the ANNs. Therefore, DL is defined as an ANN that has a larger number of parameters and layers.

The development and relationships between AI methods are shown in Fig. 2.3.

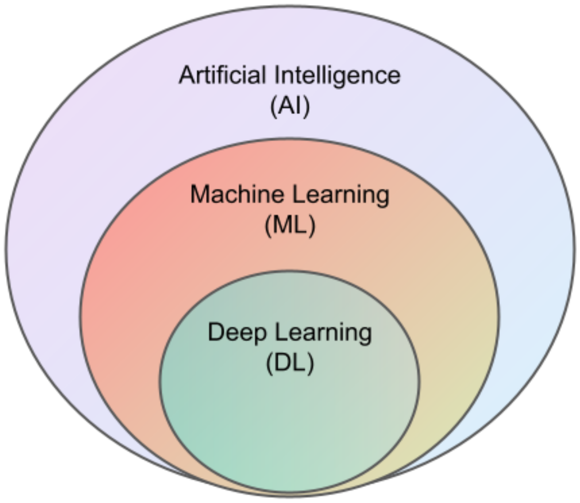


Figure 2.3: *Artificial intelligence development and expansion [2].*

This thesis focuses on ML methods. In Section 2.2, the background of traffic prediction and existing methods for traffic prediction are introduced. Section 2.3 introduces uncertainty and methods of quantifying uncertainty. In Section 2.4, the background of cancer bone and traditional methods of image recognition are demonstrated.

2.2 Methods for Traffic Prediction

Traffic congestion has become a problem in most metropolitan areas and has drawn a great deal of attention in light of traffic prediction and control systems. Therefore, the concept 'Smart City' attracts the interest of researchers and governors. The 'Smart City' is an increasingly popular concept that aims to manage the city more intelligently. The establishment of intelligent transportation systems (ITS) is one of the core tasks to build smart cities. One of the cornerstones for successfully deploying ITS is traffic speed prediction. Considering the evolution of traffic within the whole traffic network provides people and traffic managers with complete information on the traffic network to make better decisions. Traditional methods for traffic prediction are either expensive or time-consuming. Therefore, a method that can predict traffic accurately and autonomously is desired.

Mostly, the main existing traffic flow prediction approaches are divided into two methods, model-based and data-driven methods [20]. The main discrepancy between the two methods is, as they are named, whether models or data are evolved to predict the future states. Model-based methods require developing physical traffic models that describe the dynamics of the traffic network. There are mainly three types of models, such as microscopic, macroscopic and mesoscopic [21]. Microscopic models are both time and computationally expensive, as they provide details of the individual vehicle [22, 23]. Macroscopic models simulate the aggregated behaviour of traffic. However, the trade-off between computational efficiency and prediction accuracy should be compromised. Macroscopic models are used for real-time traffic prediction, such as the cell transmission model (CTM) [24] and the CTM interval [25]. By emphasising in varying levels of detail, the macroscopic models combine the microscopic and macroscopic models.

2.2.1 Segment-based Regression Kriging (SRK)

After that, model-based methods have attracted the interest of researchers. With the assumption that the models describe the dynamics of the traffic system accurately, the prediction results are reliable. Most recently, SRK [26] has been proposed for predicting traffic volume. The SRK is a geostatistical method that has been widely used for the assessment of traffic-related problems. However, most previous studies ignore critical information about the road segment, which can lead to inaccurate predictions and therefore negative impact on decision making. The SRK is proposed to address this problem. The authors make three assumptions: first, homogeneity is assumed for the traffic data over the road segment, but spatially heterogeneous and autocorrelated for different road segments; second, continuity is assumed for the observed traffic data; and third, the expected variance of the observations is assumed to be a function of separation distance, which is the spatial stationarity for the segment-based model [26]. Traffic volumes are predicted by SRK following five steps. Modelling traffic volume trends using linear regression is the first step. The Box-Cox function [27] and the selected predictor variables [26] are used to transform the latent variables. Estimating residuals is the second step after removing trends in traffic volumes. Based on the first assumption, the maximum likelihood predicted from the point values of the road segments is used to assess the segment-based variogram. The covariance between segments is formulated as:

$$\mathbb{C}_s(s_i, s_j) = \frac{1}{N(s_i)} \frac{1}{N(s_j)} \sum_{r=1}^{N(s_i)} \sum_{t=1}^{N(s_j)} \mathbf{C}(p_r, p_t) \quad (p_r \in s_i, p_t \in s_j), \tag{2.2}$$

where p_r and p_t represent discrete points within road segments s_i and s_j. The discretisation number of a road segment is represented by $N(\cdot)$. The third step is to generate the estimation and error variances. A linear combination of the surrounding road segments m is used to estimate the prediction value $\hat{z}(\cdot)$ of the SRK on the road segment s_0.

$$\hat{z}(s_0) = \sum_{i=1}^{m} W_i(s_0) z(s_i), \tag{2.3}$$

where the neighbouring segments are denoted as s_i, and the weight is denoted as $\mathbf{W}_i(s_0)$. The weights are estimated as follows:

$$\mathbf{C}_s(s_o, s_i) = \begin{cases} \sum_{j=1}^{m} W_j(s_0)\mathbb{C}_s(s_i, s_j) + \mu(s_0), & i = 1, 2, \cdots, m \\ \sum_{j=1}^{m} W_j(s_0) = 1 \end{cases}, \tag{2.4}$$

and the variance of error for SRK estimation at road segment s_0 is formulated as,

$$\hat{\sigma}^2(s_0) = \mathbf{C}_s(s_o, s_0) - \sum_{j=1}^{m} W_j(s_0)\mathbb{C}_s(s_0, s_j) - \mu(s_0) \tag{2.5}$$

Generating the final prediction by adding SRK estimation and the estimated trends in the first step is fourth step. Performance validation shows that the SRK provides a significant improvement in prediction accuracy for heavy vehicles. The best performance is provided by the point-based geostatistical model with 78% improvement in spatial variance and 53% improvement in estimated uncertainty, compared to regression kriging. This improvement shows that the SRK can provide a new perspective on characterising spatial features and spatial homogeneity of road segments.

Although, as a number of models have been developed, the general model that can satisfy all traffic scenarios is still not proposed, which limits the application of model-based methods. Data-driven methods [20] are more popular nowadays. Although model-based methods put effort into building physical models [20, 28], data-driven methods usually require only historical data. Statistical and machine learning methods are the two main categories that are developed to investigate the inherent relationships in data.

The exploration of SRK underscores the criticality of addressing the spatial characteristics and homogeneity inherent in the road segments to understand the prediction of traffic volume. While pioneering in spatially characterising traffic dynamics, this approach primarily leans toward the spatial aspect of the data. In the domain of traffic prediction, it is essential to simultaneously consider the temporal evolution of the data. Historical and sequential patterns inherent in time series data have been a cornerstone of various predictive models. A paradigm shift towards the time-series domain introduces us to the Auto Regressive Integrated Moving Average (ARIMA) model, a vanguard in temporal traffic forecasting. In the subsequent discussion, a comprehensive examination of ARIMA will be carried out, elucidating its distinctness from SRK and emphasising its robustness in capturing the temporal intricacies of traffic data.

2.2.2 Auto Regressive Integrated Moving Average (ARIMA)

The ARIMA [28] is proposed in 1979, and it achieved success on short-term highway traffic prediction. ARIMA is a popular prediction model for time series, and it consists of the autoregressive component, the integrated component, and the moving average component. The autoregressive component characterises the dependence of the current value based on past data. The integrated component is applied when the data is non-stationary, meaning that the distribution of the data varies over time. The moving average component focuses on the correlation between the current value and the past errors. The parameters for the ARIMA model are selected based on the characteristics of the data and optimised by maximising the likelihood. The mathematics of ARIMA is introduced as follows. Let $\mathbf{V}_t$ be the traffic flow time series and B be the backshift operator $B\mathbf{V}_t = \mathbf{V}_{t-1}$. As the time series $\mathbf{V}_t$ is stationary and has a mean μ, the ARIMA model is represented as:

$$\Phi_p(B)(1-B)^d(\mathbf{V}_t - \mu) = \theta_q(B)\mathbf{n}_t, \tag{2.6}$$

where $\Phi_p(B) = 1 - \Phi_1 B - \Phi_2 B^2 - \cdots - \Phi_p B^p$ is an auto regressive operator of order p, $\Theta_q(B) = 1 - \Theta_1 B - \Theta_2 B^2 - \cdots - \Theta_q B^q$ is a moving average operator of order q, and $\mathbf{n}_t \sim \mathcal{N}(0, \sigma_z^2 \mathbf{I})$ is a white noise. The ARIMA model is fitted to a specific dataset by iterative procedures: preliminary identification, estimation and diagnostic check [28]. Parameters (p, q, d) are estimated in preliminary identification by inspecting the autocorrelations of the time series. After determining the values of p, q, and d, non-linear least square techniques are used to estimate the autoregressive and moving average parameters [28], at the estimation stage. Finally, the goodness of the model is checked in the diagnostic check.

While the ARIMA model excels in capturing temporal patterns of traffic data with its methodologically robust components, its efficacy diminishes in scenarios where spatio-temporal relationships within intricate traffic networks become pivotal. Furthermore, as elucidated, machine learning techniques have gained considerable traction due to their ability to adeptly handle the spatiotemporal intricacies of traffic data. On the other hand, machine learning methods [29], [30], [31] and [32] have become increasingly popular. The spatiotemporal features of traffic networks have attracted significant interest from researchers. The latter methods, such as ANNs, have demonstrated prowess in handling multi-dimensional data streams and offering generalisable solutions [33]. Park et al. [34] proposed a real-time vehicle speed prediction algorithm based on ANN. Transitioning into the next section, an approach in forecasting, Bayesian Combined Neural Network (BCNN), will be described. This paradigm ingeniously amalgamates Bayesian principles with ANNs, furnishing a method that not only caters to the predictive strengths of neural architectures, but also accommodates the probabilistic inferences inherent in Bayes' theorem.

2.2.3 Bayesian Combined Neural Network (BCNN)

Zheng et al. [35] combined the Bayes theorem with an ANN to predict the short-term flow of freeways. The Bayesian combination approach aims to combine several predictors based on the Bayes rule and conditional probability [36]. In other words, the BCNN consists of multiple ANNs and is trained with a Bayesian approach. The prior distributions of the model parameters are specified at the beginning. These priors are updated by training, and the posterior distributions of the parameters can be estimated. BCNN combines multiple outputs generated by ANNs, and the uncertainty can be estimated by the variance of the posterior. The basic predictor is formulated for the specific t-step time series as

$$y_t = f_t^k(y_{t-1}, y_{t-2}, \cdots, y_1) + e_t^k, \tag{2.7}$$

where the target traffic flow is denoted as y_t; $f_t^k(\cdot)$ is the forecasting model; e_t^k is the corresponding prediction error; and k represents the index of predictors. However, equation (2.7) only holds for one k in each time series, and the best-fitted model cannot be identified in advance most of the time [35]. Therefore, an inducing variable Z is assumed to sample one of the values of k in one time interval to introduce uncertainty. The conditional posterior probability is therefore defined as $p_t^k = \mathbf{Prob}(Z = k/y_t, y_{t-1}, \cdots, y_1)$, and with the Bayes rule.

$$p_t^k = \frac{\mathbf{Prob}(y_t, Z = k/y_t, y_{t-1}, \cdots, y_1)}{\sum_{m=1}^{k} \mathbf{Prob}(y_t, Z = m/y_t, y_{t-1}, \cdots, y_1)} \tag{2.8}$$

From assuming that $e_t^k = y_t - f_t^k$ is a Gaussian white noise with zero mean and standard deviation σ_k and the fact of

$$\mathbf{Prob}(y_t, Z = k/y_{t-1}, y_{t-2}, \cdots, y_1) = \mathbf{Prob}(y_t/y_{t-1}, y_{t-2}, \cdots, y_1, Z = k)p_{t-1}^k, \tag{2.9}$$

and

$$\begin{aligned} \mathbf{Prob}(y_t/y_{t-1}, y_{t-2}, \cdots, y_1, Z = k) &= \mathbf{Prob}(e_t^k = y_t - y_t/y_{t-1}, y_{t-2}, \cdots, y_1, Z = k) \\ &= \frac{1}{\sqrt{2\pi\sigma_k}} e^{[-\frac{(y_t - f_t^k)}{\sigma_k}]^2} \end{aligned} \tag{2.10}$$

it can be derived that:

$$p_t^k = \frac{\frac{1}{\sqrt{2\pi\sigma_k}} p_{t-1}^k e^{[-\frac{(y_t - f_t^k)}{\sigma_k}]^2}}{\sum_{m=1}^{k} \frac{1}{\sqrt{2\pi\sigma_m}} p_{t-1}^m e^{[-\frac{(y_t - f_t^m)}{\sigma_m}]^2}}. \tag{2.11}$$

Equation (2.11) represents the probability of model k generating the observed traffic flow, and it can also be considered as the kth predictor in the combined model.

Furthermore, equation (2.11) shows that the model generating the largest prediction error $y_t - f_t^k$ is heavily penalised, and therefore a decrease p_t^k is obtained. As a result, the model that best predicts traffic flow at step t will obtain the highest p_t^k and the predictor with the highest p_t^k is the main predictor in the next step. Therefore, the prediction for the $t+1$ time step can be formulated as follows.

$$y_{t+1} = \sum_{k=1}^{k} p_t^k \cdot f_{t+1}^k. \tag{2.12}$$

Two predictors, a back-propagation NN and a radial basic function (RBF) NN, are applied. Based on equation (2.11), the posterior probability of the observed traffic flow is calculated as follows:

$$p_t^k = \frac{\frac{1}{\sqrt{2\pi\sigma_k}} p_{t-1}^k e^{[-\frac{(y_t - f_t^k)}{\sigma_k}]^2}}{\frac{1}{\sqrt{2\pi\sigma}} p_{t-1}^1 e^{[-\frac{(y_t - f_t^1)}{\sigma}]^2} + \frac{1}{\sqrt{2\pi\sigma}} p_{t-1}^2 e^{[-\frac{(y_t - f_t^2)}{\sigma}]^2}} \tag{2.13}$$

The output of the BCNN predictor at time $t+1$ is formulated as,

$$y_{t+1} = p_t^1 \cdot f_{t+1}^1 + p_t^2 \cdot f_{t+1}^2, \tag{2.14}$$

where p_t^1 and p_t^2 are the credit values for back propagation and RBF neural networks respectively. Experiments show that the proposed BCNN model outperforms the single predictor for more than 85% time steps [35].

However, an ANN cannot understand and learn the spatial relationships between the road segments. Furthermore, compared to deep learning methods, ANNs provide lower prediction accuracy because of their shallow architectures and ignorance of the time characteristics of time series inputs. Therefore, an additional temporal component is introduced into the ANNs, called recurrent neural networks (RNNs). The activations obtained from the previous layer are combined with the inputs and fed back into the RNN [37]. Liu et al. [38] proposed a state space NN (SSNN) to predicting urban travel time. A time-delay NN (TDNN) is a variant that combines previous input and current input. Shen et al. [39] prove that TDNN provides a higher accuracy in travel time prediction. Deep learning methods were first applied for traffic flow prediction

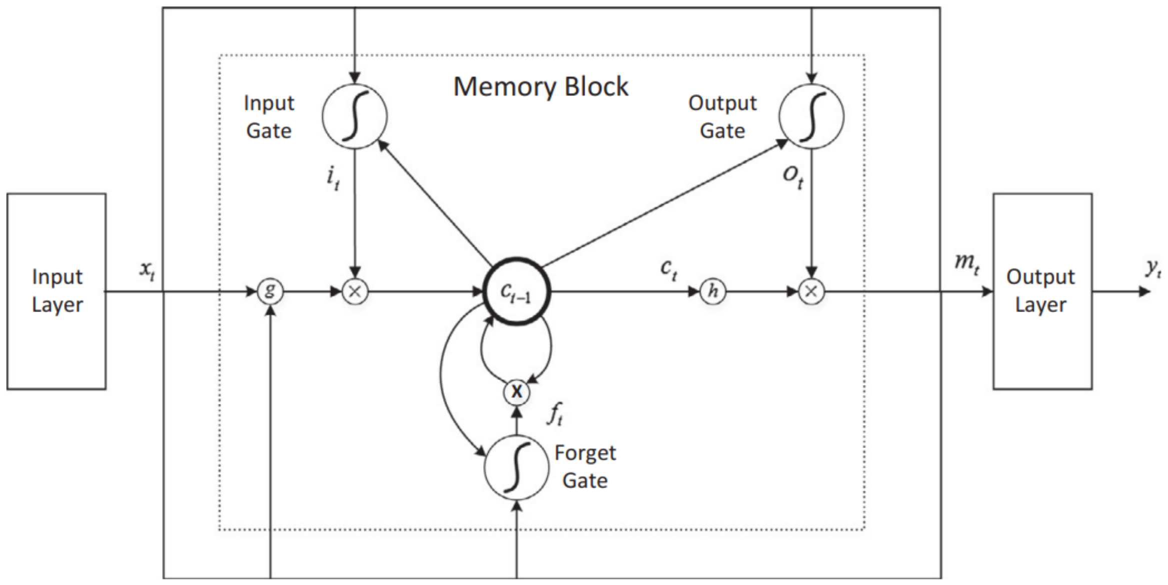


Figure 2.4: *An example of LSTM-NN architecture [3]*

by Polson and Sokolov [40]. Huang et al. [41] propose a deep belief network (DBN) for the transportation network. Ma et al. [31] proposed a combination of RNN and deep Restricted Boltzmann Machines (RBM), named RBM-RNN, which adopts the advantages of both RBM and RNN. Although RNN shows an excellent ability to solve non-linear regression problems for time series [3], time lag is an issue. The time lags commonly existing in traffic data cannot be modelled by traditional RNNs that are highly dependent on the pre-specified lag [42]. The long short-term memory neural network is proposed to address the drawbacks of RNNs in the next section.

2.2.4 Long Short-term Memory Neural Network (LSTM-NN)

Ma et al. [3] proposed a LSTM-NN and demonstrated that LSTM-NN provides outperforming stability and accuracy. An LSTM-NN contains three layers: input layer, recurrent hidden layer and output layer. Memory blocks in the hidden layers make the difference between LSTM-NN and traditional NN. The memory block is composed of self-connecting memory cells memorising the temporal states, adaptive and multiplicative gate units to control the flow of information [3]. In addition, an additional pair of input and output gates is applied to control the input and output activations flowing into the block. The core of the memory cell is the constant error carousel (CEC) [3], and the activation of CEC represents the state of the cell. The CEC enables learning of the gates, and hence the LSTM-NN is able to handle the error disappearing by maintaining it at a constant value. A forget gate is added, since it can prevent internal cell values from growing without bound, which resets the memory blocks and substitutes the CEC weights with the activation obtained from the forget gate, when the information flow is antiquated [3]. An example architecture of the LSTM-NN is presented in Fig. 2.4. In the context of traffic speed prediction, the input is denoted as $\mathbf{v} = [v_1, v_2, \cdots, v_n]$, and $\mathbf{y} = [y_{1,2}, \cdots, y_n]$ denotes the output, where n represents the prediction time step. Predicting the traffic speed in the next time step is the aim of the LSTM-NN, and the prediction can be obtained by iteratively calculating the following:

$$\mathbf{i}_n = \sigma(\mathbf{W}_{iv}\mathbf{v}_n + \mathbf{W}_{im}\mathbf{M}_{n-1} + \mathbf{W}_{ic}\mathbf{Cell}_{n-1} + \mathbf{b}_i) \tag{2.15}$$

$$\mathbf{f}_n = \sigma(\mathbf{W}_{fv}\mathbf{v}_n + \mathbf{W}_{fm}\mathbf{M}_{n-1} + \mathbf{W}_{fc}\mathbf{Cell}_{n-1} + \mathbf{b}_f) \tag{2.16}$$

$$\mathbf{Cell}_n = \mathbf{f}_t \odot \mathbf{Cell}_{n-1} + \mathbf{i}_n \odot g(\mathbf{W}_{iv}\mathbf{v}_n + \mathbf{W}_{im}\mathbf{M}_{n-1} + \mathbf{b}_c) \tag{2.17}$$

$$\mathbf{o}_n = \sigma(\mathbf{W}_{ov}\mathbf{v}_n + \mathbf{W}_{om}\mathbf{M}_{n-1} + \mathbf{W}_{oc}\mathbf{Cell}_n + \mathbf{b}_o) \tag{2.18}$$

$$\mathbf{M}_n = \mathbf{o}_n \odot h(\mathbf{Cell}_n) \tag{2.19}$$

$$\mathbf{y}_n = \mathbf{W}_{ym}\mathbf{M}_n + \mathbf{b}_y \tag{2.20}$$

where the scalar product of two vectors is denoted as $\odot$, and $\sigma(\cdot)$ is defined as,

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.21}$$

As presented in Fig 2.4, the memory block is bounded by a dished box, and the outputs of the input, output and forget gates are respectively $\mathbf{i}_n$, $\mathbf{o}_n$ and $\mathbf{f}_n$. Weight and bias are denoted as $\mathbf{W}_{(\cdot\cdot)}$ and $\mathbf{b}_{(\cdot)}$, and $\mathbf{C}_n$ and $\mathbf{M}_n$ represent the activations obtained from the cell and the memory block. For example, the weight for the input gate and applied to the input is represented as $\mathbf{W}_{iv}$. In equation (2.17), $g(\cdot)$ is the centred logistic sigmoid function with a range of $[-2, 2]$:

$$g(x) = \frac{4}{1 + e^{-x}} - 2, \tag{2.22}$$

and a centered logistic sigmiod function with range $[-1, 1]$ is denoted as:

$$h(x) = \frac{2}{1 + e^{-x}} - 1. \tag{2.23}$$

The LSTM-NN is trained based on backpropagation through time (BPTT) and real time recurrent learning with gradient descent optimisation [43, 44]. The common loss function is sum square errors. A comparative study has been done by Ma et al. [3], and the stability and accuracy of the LSTM-NN outperform when compared to RNN, TDNN, SVM, ARIMA and Kalman filter. Having discussed the architectural and operational intricacies of the LSTM-NN and its merits in traffic speed prediction, it is essential to explore other neural network architectures that have also been employed in the domain of traffic prediction. One such architecture that has garnered significant attention is CNN. While LSTM-NNs outperform in capturing long-term dependencies by leveraging their memory cells, CNNs, traditionally used in image processing tasks, have unique capabilities in automatically and adaptively learning spatial hierarchies of features. The subsequent section delves into the principles and applications of CNNs in the context of traffic prediction, elucidating its potential advantages.

2.2.5 Convolutional Neural Network for Traffic Prediction

Compared to ANN related methods, the deeper and more complex features are abstracted by deep learning methods, and thus deep learning can learn the data better than existing ANNs. However, both ANN and existing deep learning methods are focused on one road segment or a small traffic network. Most existing models only considered traffic evolution in the aspect of temporal relationships, and spatial correlations of the traffic network are not considered. Therefore, to fill the gap, Ma et al. [45]

introduced an image-based method that represents traffic data as images and applies a CNN to extract spatio-temporal features of traffic data in the form of images.

2.2.5.1 Traffic Data as Images

Traffic prediction should be investigated in both time and space dimensions. Let the x- and y-axes of a matrix represent time and space, respectively. The intervals in the time dimension depend on the sampling resolution of the data collection sensor and span from the beginning of the day to the end of the year. For instance, a conventional GPS device takes a sample every 10 seconds, while a magnetic loop sensor takes a 15-minute sample of traffic speed. The sensors are represented as dots with positions, traffic speeds, etc. in space. Since the sequences of the dots are redundant and there are many locations that lack variability, simply organising these dots by sensor IDs and fitting them onto the y-axis may cause a high dimension and an uninformative issue. To ensure that the y-axis is informative, the dots are divided into parts that reflect comparable traffic situations. Each element in the matrix is the traffic speed value associated with time and space. The matrix can be considered as a one-channel image. Finally, a spatio-temporal matrix can be constructed as following,

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{NM} \end{bmatrix}, \tag{2.24}$$

where N is the number of the time steps and M is the number of the sensors. The i-th row of $\mathbf{S}$ represents the traffic speed at the time step i. The pixel value of s_{ij} represents the traffic speed on the road segment j at the time step i. Therefore, the matrix $\mathbf{S}$ forms a single channel image. Figure 2.5 demonstrates the transformation among the raw traffic speed network, the spatio-temporal matrix, and the final image.

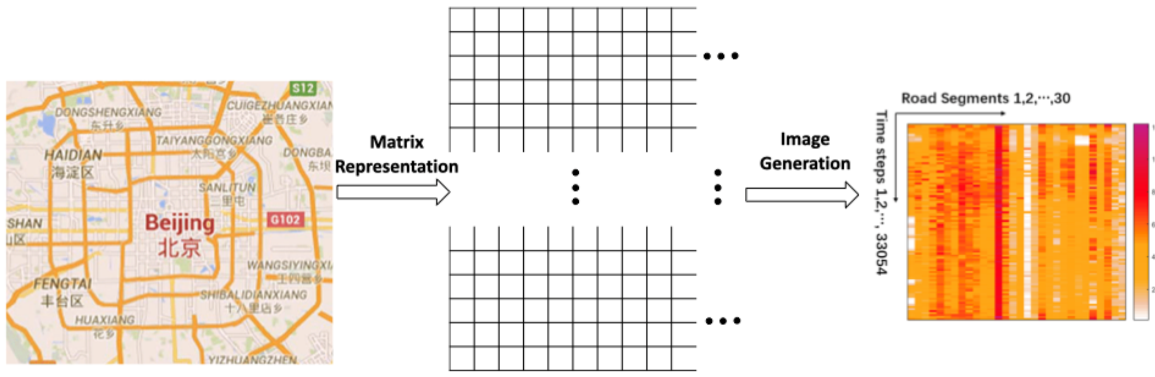


Figure 2.5: *The illustration of the traffic-to-image conversion [4].*

2.2.5.2 Convolutional Neural Network for Traffic Prediction

CNN has demonstrated a great capacity to comprehend images, due to its distinctive ways of extracting characteristics. Two features that add to the CNN's distinctiveness are the local linked layers (convolutional layers) and the pooling technique (pooling layer). The local connected layers limit the amount of parameters that must be learnt while maintaining the most crucial characteristics, since the output neurons of each layer are only linked to their nearby input neurons, which extract a new feature in every layer.

Based on the two characteristics, Ma et al. [45] introduced four major modifications for CNN in traffic speed prediction: a) the inputs are different. In case of image classification tasks, the input is usually images that have three channels and pixel values from 0 to 255. However, the CNN input in the traffic speed prediction context has only one channel and ranges from 0 to some speed values. Although there are differences, the traffic speed data are normalised to avoid difficulties in the training procedures. b) The model outputs are different from these in a classification task, whereas, the outputs of traffic speed prediction are predicted traffic speed on the road segments rather than the classification labels. c) Abstracted features have different meanings. The features can be edges or physical shapes of the training objects for image classification problems. For traffic speed prediction, the extracted features are the correlations of traffic speed within the traffic network. d) The training objectives are different. A continuous cost function, rather than a cross-entropy cost function for classification, should be adopted for traffic speed prediction, as traffic speed values are continuous.

2.2.5.3 CNN Characteristics

The architecture of the typical CNN in the context of a traffic network consists mainly of four parts that are network input, feature extraction, regression, and model output. Each component is explained below.

First, the inputs of the CNN are the images converted from a traffic network with spatio-temporal features. Let the length of the input be D and the number of the time interval be H. The matrix form of traffic speed data is the following,

$$\mathbf{S}^i = \begin{bmatrix} S_{11} & ... & S_{1D} \\ ... & ... & ... \\ S_{H1} & ... & S_{HD} \end{bmatrix}, \tag{2.25}$$

where i represents the input index, D is the number of the sensors and H is the length of the time intervals. Here V_{HD} represents the traffic speed value at the H-th time step on the D-th road segment.

Second, feature extraction is a sequence of convolutional and pooling layers. The layer index is denoted by l, and the pooling operation is represented by using **pool**. Respectively, $\mathbf{S}_j^l$, α_j^l and $(\mathbf{W}_j^l, b_j^l)$ denote the l-th layer inputs, outputs and network parameters, where j is the channel index of the convolutional filters. The number of convolutional filters in the l-th layer is denoted by C_l. The first sequence of convolutional and pooling layers can be formulated as:

$$\alpha_j^1 = \mathbf{pool}(\phi(\mathbf{W}_j^1\mathbf{S}_j^1 + b_j^1)), j \in [1, C_1], \tag{2.26}$$

where ϕ is the activation function. The output in the l-th layer can be written as:

$$\alpha_j^l = \mathbf{pool}(\phi(\sum_{k=1}^{C_l-1}(\mathbf{W}_j^l \mathbf{S}_k^l + b_j^l)), j \in [1, C_l]. \tag{2.27}$$

There are three characteristics of the extracted traffic speed features: 1) Convoltuional and pooling operations are processed in two dimensions, and therefore the CNN is able to lean the spatio-temporal relations among road segments and along time.; 2) Different depth of the network can be applied according to the circumstance by modifying the number of layers; 3) the CNN transforms the input images into deep features through a sequence of layers.

In model prediction, the features obtained from a sequence of feature extraction layers are fed into a flatten layer that generates a dense vector with the final and highest-level features of the traffic network. The densed vector can be represented as:

$$\alpha_{flatten}^l = \mathbf{flatten}([\alpha_1^l, \alpha_2^l, \cdots, \alpha_j^l]), j \in [1, C_l], \tag{2.28}$$

where l is the depth of the network and $flatten$ is the concatenating procedure.

The final step is to feed the dense vector through a fully connected layer. The model output can be written as:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{W}_f \alpha_{flatten}^L + b_f, \\ &= \mathbf{W}_f(flatten(\mathbf{pool}(\phi(\sum_{k=1}^{C_L-1}(\mathbf{W}_j^L \mathbf{S}_k^L + b_j^L)))) + b_f\end{aligned} \tag{2.29}$$

where $\mathbf{W}_f$ and b_f are the layer parameters of the fully connected layer and $\hat{\mathbf{y}}$ is the predicted traffic speed.

2.2.5.4 Convolutional and Pooling Layers

The CNN is different from traditional ANNs, as neurons are not fully connected to the output neurons. The convolutional layer in the CNN abstracts local correlations and connects only local neurons to the output neurons. One filter can extract one traffic feature, and thus the number of features abstracted depends on the number of convolutional filters. The convolutional operation provides a local path for connecting lower-level features to high-level ones. The convolutional filter $\mathbf{W}^l_{C_l}$ is applied to the input, and the output of the convolutional operation can be formulated as:

$$\mathbf{y}_{conv} = \sum_{a=1}^{m}\sum_{b=1}^{n}(\mathbf{W}^l_{C_l})_{ab}\mathbf{x}_{ab}, \tag{2.30}$$

where two dimensions of the convolutional filter are represented by m and n. The traffic speed value at position a and b is denoted as $\mathbf{x}_{ab}$, and $(\mathbf{W}^l_{C_l})_{ab}$ is the weight of the convolutional filter. The convlutional layer output is represented as $\mathbf{y}_{conv}$.

Due to the extraction of crucial values in the specific area, the pooling layer down-samples and aggregates data. Pooling operations ensure that CNN is locally invariant. Therefore, the identical features can be abstracted by the CNN, even with shift scales [46]. To conclude, reducing the network scale is one advantage of the pooling operation, but the most important aspect is to identify the most prominent features in the input. For example, maximum pooling is formulated as:

$$\mathbf{y}_{pool} = max(\mathbf{x}_{a,b}), a \in [1 \cdots p], b \in [1 \cdots q], \tag{2.31}$$

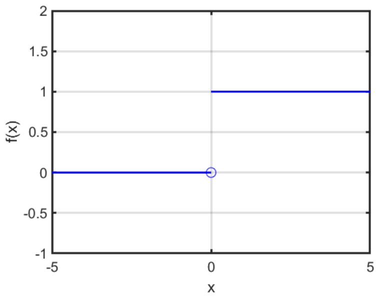
where the two dimensions of pooling filter are denoted as p and q. The traffic speed value at position a and b is represented as $\mathbf{x}_{ab}$, and $\mathbf{y}_{pool}$ is the output.

2.2.5.5 Activation Functions

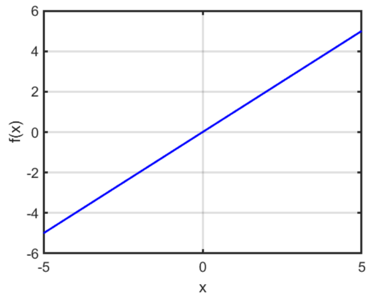
The following subsection presents the commonly used activation function [47, 48].

a) Binary Step Function. The binary step function activates the artificial neurons according to a threshold. If the input value is greater than the threshold, the neuron is activated. Otherwise, the neuron is deactivated. The mathematical definition with threshold of 0 is given as following,

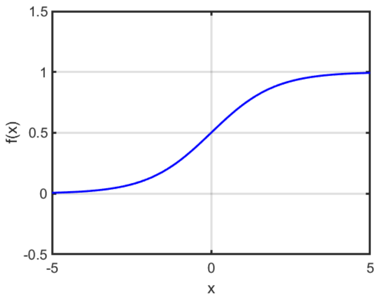
$$f(x) = \begin{cases} 0, & \textbf{if} \quad x < 0 \\ 1, & \textbf{if} \quad x \geq 0 \end{cases}, \tag{2.32}$$



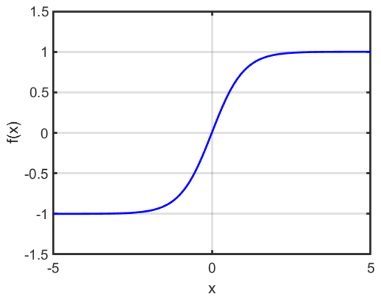
(a)



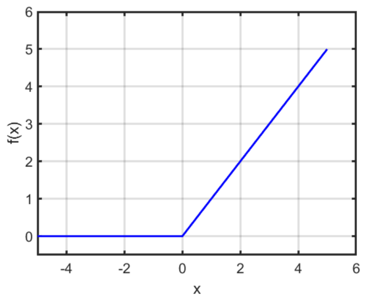
(b)



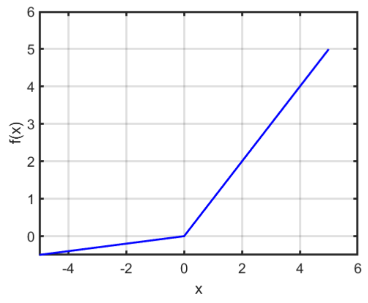
(c)



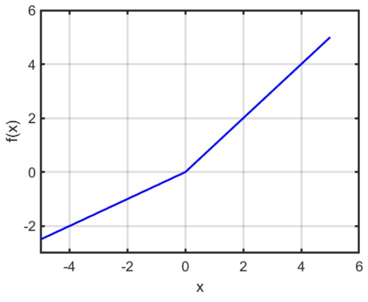
(d)



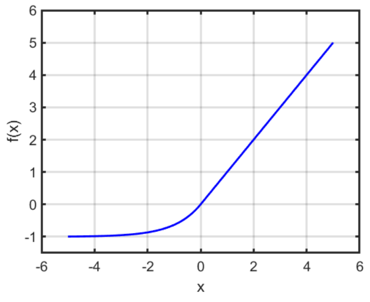
(e)



(f)



(g)



(h)

Figure 2.6: *a) Binary step activation function. b) Linear activation function. c) Sigmoid activation function. d) Tanh activation function. e) ReLu activation function. f) Leaky ReLu activation function. g) Parametric ReLu activation function. h) Exponential linear units activation function.*

where x is the input. An example diagram of the binary step function is given in Fig. 2.6 (a). There are two limitations of binary step function: 1) the binary step function cannot be applied on multi-output model, such as multi-class classification; 2) zero gradient of the binary step function is a problem for backpropagation.

b) Linear Activation Function The linear activation function is proportional to the input, which is also called the identity function. It can mathematically be represented as

$$f(x) = x. \tag{2.33}$$

An example diagram of the binary step function is given in Fig. 2.6 (b). However, the linear activation function cannot be applied on backpropagation, since the gradient is constant. On the other hand, once only the linear activation is applied, the last layer can essentially be a linear combination of the previous layers. Therefore, the network will be considered as a single layer.

c) Sigmoid Function The sigmoid function scales the input values into a range of 0 to 1. The output value will be close to 1, as the input value becomes more positive. The output value will be closer to 0, as the input value becomes more negative, as shown in Fig. 2.6 (c). The mathematical definition is given as

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2.34}$$

The sigmoid function is commonly applied when the output of the network is probabilities. Since the probability is within the range of 0 to 1, the sigmoid function is a right choice because of its range.

However, Fig. 2.7 implies that the gradient of the sigmoid function is very small when the input value is greater than 3 or smaller than -3. As the gradient approaches zero, the network will be affected by the vanishing gradient problem.

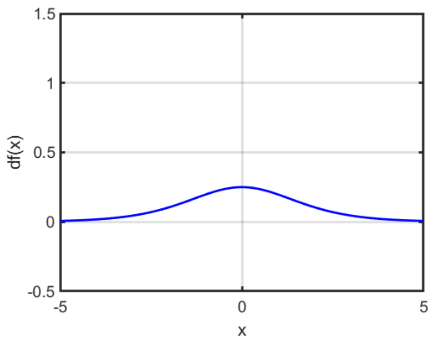


Figure 2.7: *The derivative of the Sigmoid Activation Function.*

d) Tanh Function. As shown in Fig. 2.6 (d), the tanh function has the same S-shape as the sigmoid function. The sigmoid function scales the input values in a range of -1 to 1. The output value will be close to 1, as the input value becomes more positive. The output value will be close to -1, as the input value becomes more negative. The tanh function can be mathematically defined as,

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.35}$$

As shown in Fig. 2.6 (d), the tanh function is zero-centred, and hence, the output can be easily mapped to strongly positive, neutral or strongly negative. However, the gradient curve of the tanh function is slimier as that of sigmoid. Therefore, the tanh function also faces the problem of vanishing gradients.

e) Rectified Linear Unit (ReLu) Function Unlike the linear activation function, the ReLu function only activates neurons when the input is greater than 0. The mathematical definition is given as following,

$$f(x) = \max(0, x). \tag{2.36}$$

Although the diagram, shown as Fig. 2.6 (e), of the ReLu function is identical when the input is greater than 0, it has a derivative function and allows backpropagation. The advantages of the ReLu function are as follows: 1) the ReLu function takes less computational resources compared with sigmoid and tanh functions, since only part of the neurons are activated; 2) because of the linear and non-saturating property of the ReLu function, the loss function converges more efficiently. However, the Dying ReLU problem occurred. The weights and biases in the neurons that generate negative values are never updated during backpropagation, as the gradient in these neurons is 0, which leads to the problem that some neurons are never activated. Therefore, it reduces the ability of the network to fit the data correctly.

f) Leaky ReLu Function Leaky ReLu function is a variant version of the ReLu function to handle the problem existing with the ReLu function. Instead of 0, the leaky ReLu function outputs a small slope when the inputs are negative. The diagram is shown in Fig. 2.6 (f). The mathematical definition is shown as following,

$$f(x) = \max(0.1x, x). \tag{2.37}$$

In addition to the advantages of the ReLu function, a leaky ReLu function also allows backpropagation for negative inputs, as the gradient of negative inputs is a non-zero value. However, the limitation is that the small value of the gradient over the negative input makes learning the parameters time-consuming.

g) Parametric ReLu Function Parametric ReLu function is another variant of the ReLu function to solve the dying ReLU problem. The difference between leaky and parametric ReLu functions is that the coefficient of the parametric ReLu function for the negative inputs is a parameter. It can be mathematically defined as,

$$f(x) = \max(ax, x). \tag{2.38}$$

h) Exponential Linear Units (ELU) Function The ELU function is another improved version of the ReLu function. The slope of the negative input is modified as a function of the ELU function. The mathematical definition is given as

$$f(x) = \begin{cases} x, \quad \textbf{if} \quad x \geq 0 \\ \\ \alpha(e^x - 1), \quad \textbf{if} \quad x < 0 \end{cases}. \tag{2.39}$$

Although the ELU function is a strong alternative for the ReLu function, the ELU function has the following limitations: 1) the computational complexity increases because of the exponential operation; 2) the learning process of the parameter α is not taken.

2.2.5.6 CNN Optimisation

Traffic speeds in the complex traffic network are the output of CNN, and the loss function of the network is the mean squared errors (MSEs), which measure the error between prediction and ground-truth traffic speeds. Therefore, the aim of optimisation is to minimise MSEs during training procedures. MSE is formulated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2. \tag{2.40}$$

The model parameters are denoted as $\theta = (\mathbf{W}_i^l, b_i^l, \mathbf{W}_f, b_f)$, backpropagation algorithm [49, 46] is applied to optimize the values of θ:

$$\theta = argmin_{\Phi}\frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2. \tag{2.41}$$

2.2.6 Structural Recurrent Neural Network for Traffic Prediction

The CNNs have been reported to achieve effective forecasting accuracy for traffic speed [45]. CNNs are proven to be powerful in learning spatio-temporal features where the traffic speed data are converted into spatio-temporal images. The successful combination of convolutional and pooling layers enables the model to capture high-order features in the input data. Recurrent neural networks (RNNs) have been considered, since the forecasting of traffic speed data is essentially a time series prediction. In this section, the architecture of structural-RNN is described [5]. It starts with an introduction of the representations of spatio-temporal graphs (ST-graphs). The RNN, therefore, can be represented by the factor component decomposed from an ST-graph.

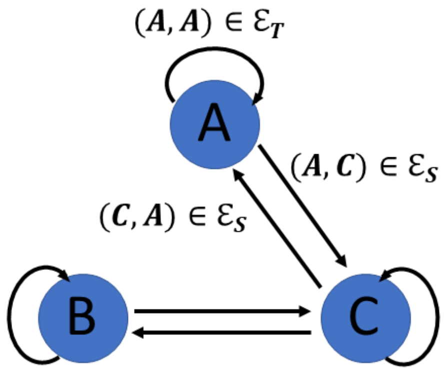
2.2.6.1 Traffic Speed Data as Time Series

The same as [45] introduced, the model aims to predict the short-term traffic speed data by using the road network ST-graph and the historical traffic speeds. Let v_n^t represent the speed of traffic on the road segment n at time t. Given a sequence of traffic speed data $\mathbf{v}_n^t$ for the road segment $n = 1, 2, \cdots, N$ at time steps $t = T - l, \cdots, T$, the model is aimed at forecasting the future traffic speed $\mathbf{v}_n^{T+l}$.

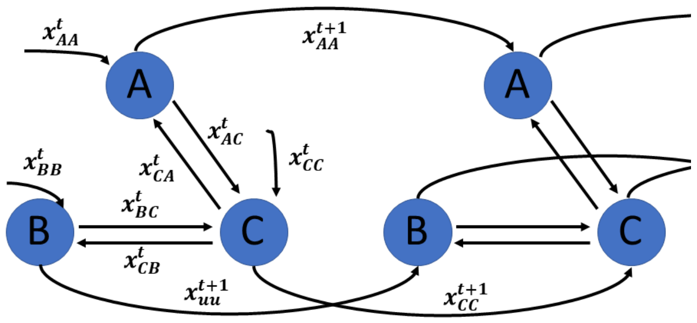
2.2.6.2 Structural RNN Architectures

In this section, the architecture of the structural RNN is introduced. It starts with the introduction of the spatio-temporal (ST) graphs. Then the ST-graph is decomposed into factor components, and each factor is represented by using an RNN. By following the architecture and interactions of the ST-graph, the RNNs are interconnected.

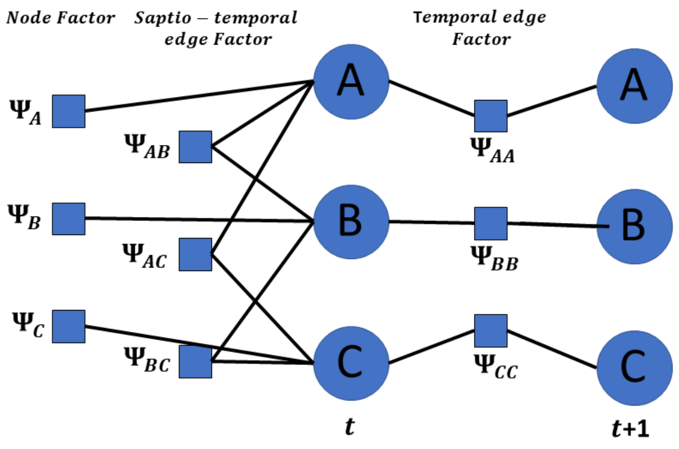
a) Representation of spatial-temporal graphs: An ST-graph is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$, where the nodes are denoted as $\mathcal{V}$, and the spatial and temporal edges are represented as $\mathcal{E}_S$ and $\mathcal{E}_T$ respectively [50]. In the traffic speed prediction context, the nodes in the ST-graph are assigned to be the road segments in the traffic network. The information of correlations between neighboring road segments is contained in the spatial edges, and the dynamics of the traffic evolution over time are represented by the temporal edges. An example of the ST-graph is presented in Fig. 2.8 a), where the nodes $A, B, C \in \mathcal{V}$ represent road segments, and the links between different nodes represent the spatial edges $\mathcal{E}_S$. Spatial edges exist only if there are connections between two road segments. The directions are under consideration, and thus two spatial edges in opposite directions are applied if neighbouring nodes are connected in both directions. The nodes at adjacent time steps are connected by temporal edge connections, such as the nodes at time t and $t + 1$ [50]. In Fig. 2.8 b), the traffic speed at node A at time t is represented as x_A^t, and the feature vector of the spatial edge connecting nodes A and B at time t is represented as $x_{AB}^t = [x_A^t, x_B^t]$. The feature vectors of two spatial edges in the opposite direction that connect nodes A and B are different,


$(A, A) \in \mathcal{E}_T$
$(A, C) \in \mathcal{E}_S$
$(C, A) \in \mathcal{E}_S$
A
B
C

(a)



(b)



(c)

Figure 2.8: *An example of ST-graph. a) Nodes A, B and C represent the road segment, and are connected by spatial edges $\mathcal{E}_S$ and temporal edges $\mathcal{E}_T$. b) The ST-graph evolves over time. The unrolling is represented by using temporal edges $\mathcal{E}_T$. c) The factor graph representation of the ST-graph. Nodes and edges in the ST-graph are represented as factors [5].*

such as $x_{AB}^t = [x_A^t, x_B^t]$ and $x_{BA}^t = [x_B^t, x_A^t]$. The feature vector of the temporal edges connecting the node itself from a previous time is represented as $x_{AA}^t = [x_A^{t-1}, x_A^t]$.

For a given ST-graph and feature vectors, the goal is to obtain the prediction of the node outputs y_A^t, where $A \in \mathcal{V}$. The features at the node and the interactions with the connecting nodes are two factors that have an impact on the speed of traffic y_A^t [50], which forms a complex system. The interactions with the connecting nodes are formulated to a simpler function using a factor graph [51]. The factor graph can be formulated from the st graph, and thus the RNN can be derived from the specific factor graph. Two factor functions are included in the factor graph, $\Psi_A(y_A, x_A)$ and a pairwise factor $\Psi_e(y_{e(1)}, y_{e(2)}, x_e)$, where $A \in \mathcal{V}$ represents the node and $e \in \mathcal{E}_S \cup \mathcal{E}_T$ represents the edges. Fig. 2.8 c) shows the factor-graph representation of the ST-graph in Fig. 2.8 a).

Learning each factor in the factor graph is necessary. However, it is unnecessary to learn distinct parameters for every node, but the parameters and factors can be shared by similar nodes. For example, nodes B and C can share factors and parameters when the traffic speed of A is predicted. Sharing the factors and parameters provides the flexibility to process an increasing number of nodes without increasing the computational complexity. Therefore, the nodes are partitioned as $\mathcal{C}_{\mathcal{V}} = \mathcal{V}_1, \cdots, \mathcal{V}_P$, where $\mathcal{V}_p$ represents similar nodes. The factor $\mathcal{V}_p$ shared is denoted as $\Psi_{\mathcal{V}_p}$. The partition of the semantically similar nodes naturally divides the edges. Thus, the edges are partitioned as $\mathcal{C}_E = E_1, \cdots, E_N$, where E_n are the edges of similar nodes and all edges in E_n share the same factor Ψ_{E_m}. The node factor $\Psi_{\mathcal{V}_p}$ and the edge factors Ψ_{E_n} connected to node A are considered to predict the traffic speed. The structured RNN is established by using this definition such that the interactions within the ST-graph can be captured.

b) Structural RNN from Spatio-temporal Graphs: the architecture represents each factor by using an RNN. Three types of RNNs are defined for node, spatial and temporal edges, which are nodeRNNs, spatial and temporal edgeRNNs. The nodeRNNs are denoted as $\mathbf{R}_{\mathcal{V}_p}$. The spatial and temporal RNNs are represented as $\mathbf{R}_{\mathcal{E}_S}$ and $\mathbf{R}_{\mathcal{E}_T}$

respectively. The combination of spatial and temporal edgeRNNs refers to edgeRNNS $\mathbf{R}_{E_m}$. Interactions between connecting nodes are captured by jointly considering both nodeRNNs and edgeRNNs. To form a feedforward network, nodeRNNs and edgeRNNS are connected to form a bipartite graph $\mathcal{G}_R = (\mathbf{R}_{\mathcal{V}_p}, \mathbf{R}_{E_m}, \mathcal{E}_R)$ [50]. In particular, nodeRNNs and edgeRNNs are connected, **iff** their factors $\Psi_{\mathcal{V}_p}$ and Ψ_{E_m} are connected in the ST-graph.

Algorithm 1 Spatio-temporal graph to structural RNN [50]

```
Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$, $\mathcal{C}_{\mathcal{V}} = \mathcal{V}_1, \cdots, \mathcal{V}_P$
Output: Structural RNN graph $\mathcal{G}_R = (\mathbf{R}_{\mathcal{V}_p}, \mathbf{R}_{E_m}, \mathcal{E}_R)$
1: Semantically partition edges $\mathcal{C}_E = E_1, \cdots, E_N$
2: Find factor components $\mathcal{V}_p, \Psi_{E_m}$ of $\mathcal{G}$
3: Represent each factor as nodeRNNs $\mathbf{R}\mathcal{V}_p$ and edgeRNNS $\mathbf{R}_{\mathcal{E}_S}$
4: Connect $\mathbf{R}_{\mathcal{V}_p}$ and $\mathbf{R}_{E_m}$ for a bipartite graph.
5: return $\mathcal{G}_R = (\mathbf{R}_{\mathcal{V}_p}, \mathbf{R}_{E_m}, \mathcal{E}_R)$
```

The **Algorithm** 1 shows the procedures of constructing a structural RNN architecture. The predictions of nodeRNNs interact through the edgeRNNs, and therefore, the prediction of the node is obtained. In the next section, a training procedure of structural RNN is introduced.

c) Structural-RNN Architecture Training: in order to train the structural-RNN, a sequence of features $\{\mathbf{x}_A^t\}_{t=T-l+1}^{T}$ associated with each node in the ST-graph is fed into the structural-RNN architecture for a given node $A \in \mathcal{V}_p$. Once the node features input, the structural-RNN architecture is supposed to predict the traffic speed y_A^{t+1} at the node. The edge features on edge $e \in E_m$, ordering in time, is the input of the edgeRNNs,

The temporal sequence of edge features $\mathbf{x}_e^t$ on edge $e \in E_m$ is input into the edge node edgeRNNs, and the edge is the events at the node A in the ST-graph [50]. In the case of the prediction of traffic speed, the edge e includes spatial and temporal

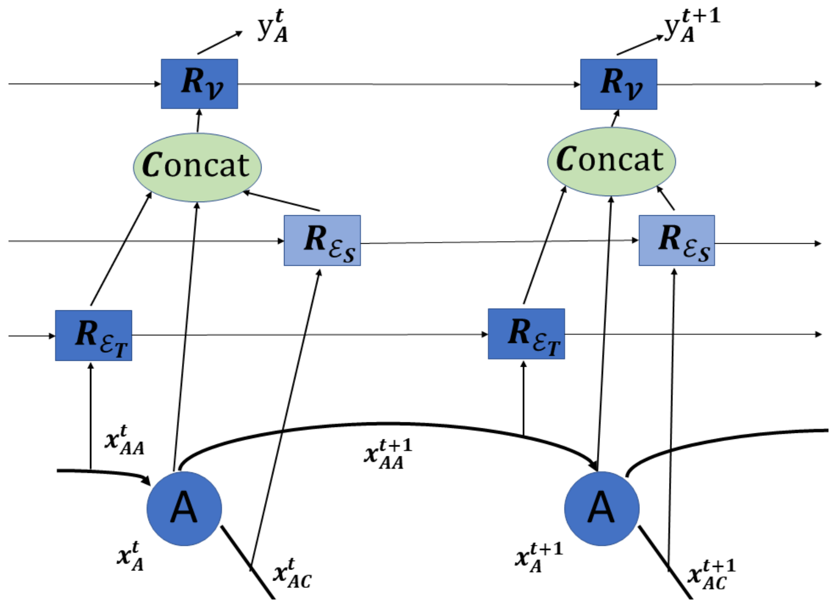


Figure 2.9: *The architecture of the structural-RNN of specific node A with the unrolled spatio-temporal graph [5].*

edges. The connected output of edge RNNs is concatenated to the node features $\mathbf{x}_A^t$ in nodeRNNs at each time step. During training, predictions error is backpropagated through nodeRNNs and edgeRNNs. In this way, the node and edge features of the node are non-linearly combined to form a structural-RNN.

Fig 2.9 shows the architecture of structural-RNN at the specific node A. Forward unrolling involves the edgeRNNs $\mathbf{R}_{\mathcal{E}_T}$ and $\mathbf{R}_{\mathcal{E}_S}$. NodeRNNs $\mathbf{R}_{\mathcal{V}}$ combines the node features and the output of edgeRNNs to form the prediction of traffic speed at each step.

The model-based and data-driven methods introduced above are considered as parametric model, since they consist of a specific set of parameters. Unlike the parametric models that assume a specific prior, non-parametric models are more flexible and able to adapt when data changes over time, which makes them more robust. The non-parametric models are a potential solution to the overfitting problem that occurs with a parametric model, because of their flexibility. On the other hand, the non-parametric models handle data with small sample size and missing data more effectively, since they learn the distributions of parameters rather than specific values. The next two sections introduce two non-parametric models that have been applied for traffic prediction.

2.2.7 Support Vector Machines

Statistical data-driven methods, such as SVM, have achieved success in time series data. Zhang et al. [52] proposed a least squares SVM (LS-SVM) for traffic forecasting. LS-SVM replaces quadratic programming (QP) with linear least squares criteria [53]. Let the dataset be $D = (\mathbf{x}_i, y_i)_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the input and $y_i \in \mathbb{R}$ is the target. The LS-SVM can be formulated as following,

$$y_i = \mathbf{W}^T \phi(\mathbf{x}_i) + b + e_i, \quad i = 1, 2, \cdots, n, \tag{2.42}$$

where $\phi(\cdot)$ is a non-linear mapping function; the weight is denoted as $\mathbf{W} \in \mathbb{H}$; e_i is an error term; and b is the bias term. Therefore, the optimisation problem is formulated in the weight space as following,

$$\min \mathcal{L}(\mathbf{W}, \mathbf{e}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}\Gamma \sum_{i=1}^{n} e_i^2, \tag{2.43}$$

where the loss function is represented as $\mathcal{L}$, and the error coefficient of the weighted least squares cost function is denoted as Γ [52]. The Lagrangian function is defined for solving the minimisation problem,

$$\mathcal{L}(\mathbf{W}, b, \mathbf{e}, \alpha) = \mathcal{L}(\mathbf{W}, \mathbf{e}) - \sum_{i=1}^{n} \alpha_i [\mathbf{W}^T \Phi(\mathbf{x}_i) + b + e_i - y_i], \tag{2.44}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_n] \in \mathbb{R}^n$ is the support vector. The gradient conditions for optimality are given by:

$$\begin{cases} \frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{w}^T \Phi(\mathbf{x}_i), \\ \frac{\partial L}{\partial b} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i = 0, \\ \frac{\partial L}{\partial e_i} = 0 \Rightarrow \alpha_i = \Gamma e_i, \\ \frac{\partial L}{\partial \alpha_i} = 0 \Rightarrow \mathbf{w}^T \Phi(\mathbf{x}_i) + b + e_i - y_i = 0. \end{cases} \tag{2.45}$$

For standard SVM, $\mathbf{W}$ and $\phi(x_i)$ are not calculated [54]. A linear system can be obtained by eliminating $\mathbf{W}$ and $\mathbf{e}$:

$$\begin{bmatrix} 0 & \mathbf{I}^T \\ \mathbf{I} & \boldsymbol{\Omega} + \frac{1}{\Gamma}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \tag{2.46}$$

where $\mathbf{y}$ is the targets, $\mathbf{I}$ is the identical matrix, and $\Omega_{i,j} = \phi(x_i)^T \Phi(x_j)$. According to the Mercer condition [53], a mapping $\phi(\cdot)$ and an expansion $\mathbf{K}(x, y) = \sum_i \phi(x)\phi(y), x, y \in \mathbb{R}^n$ exist, if and only if $\int \mathbf{K}(x, y)g(x)g(y)dxdy \geq 0$ with $\int g(x)^2 dx$ finit for any $g(x)$. Therefore,

$$\Omega_{i,j} = \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j) = \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j). \tag{2.47}$$

The LS-SVM model can be formulated as following,

$$f(\mathbf{x}) = \mathbf{W}^T \phi(\mathbf{x}_i) + b = \sum_{i=1}^{n} \alpha_i \mathbf{K}(\mathbf{x}, \mathbf{x}_j) + b, \tag{2.48}$$

where solving the equation (2.46) provides the values of $\boldsymbol{\alpha}$ and b. For a positively defined kernel function, there are three commonly used choices, such as the squared exponential kernel, which are introduced in Section 2.2.8. The LS-SVM provides a non-parametric model for traffic prediction and achieves better performance compared with ARMA, Kalman filter (KF), historical-mean (HM) and radial basis function neural network (RBF-NN).

2.2.8 Gaussian Process Methodology

The previous section introduces powerful parametric methods such as CNNs and RNNs. In this section, GP, another machine learning method different from neural networks, is introduced. NNs provide deterministic predictions whereas the GPs are considered as a non-parametric model and provide additional uncertainty information. In addition, DNN is equivalent to a GP when the network is infinitely wide or deep [55]. The advantages mentioned above motivate the research and methods proposed in this thesis. This section briefly introduces GP regression in mathematics.

2.2.8.1 Gaussian Process Regression

a GP is defined as a collection of random variables, with any finite number of which has a joint Gaussian distribution [56]. The aim of a GP is to model the unknown function f. Let $\mathbf{x}$ and $\mathbf{y}$ be, respectively, the input and output of a GP, and they can be multidimensional. The GP is a non-parametric Bayesian model, which places a GP prior over the latent function f [57] shown as following

$$f(\mathbf{s}) \sim \mathcal{GP}(m(\mathbf{s}), K(\mathbf{s}, \mathbf{s}')), \tag{2.49}$$

where $m(\mathbf{s})$ and $K(\mathbf{s}, \mathbf{s}')$ denote the mean and covariance function that is symmetric positive semi-definite, and all possible pairs of the input are dented as $\mathbf{s}$ and $\mathbf{s}'$. In the function-space view, the unknown function is Gaussian distributed.

$$f(\mathbf{s}) \sim \mathcal{N}(m(\mathbf{s}), K(\mathbf{s}, \mathbf{s}')), \tag{2.50}$$

$$\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix}, f(\mathbf{s}) = \begin{bmatrix} f(s_1) \\ f(s_2) \\ \vdots \\ f(s_N) \end{bmatrix}, m(\mathbf{s}) = \begin{bmatrix} m(s_1) \\ m(s_2) \\ \vdots \\ m(s_N) \end{bmatrix} \tag{2.51}$$

$$K(\mathbf{v}, \mathbf{s}') = \begin{bmatrix} k(s_1, s_1') & k(s_1, s_2') & \cdots & k(s_1, s_N') \\ k(s_2, v_1') & k(s_2, s_2') & \cdots & k(s_2, s_N') \\ \vdots & \vdots & \ddots & \vdots \\ k(s_N, s_1') & k(s_N, s_2') & \cdots & k(s_N, s_N') \end{bmatrix}, \tag{2.52}$$

where the output vector for the N-dimensional input $\mathbf{s}$ is represented as $f(\mathbf{s})$, and the multivariate normal distribution is denoted as $\mathcal{N}(\cdot, \cdot)$. Equation (2.52) shows a general form of the covariance function, and thus the covariance function can be alternatively denoted as $\mathbf{K}_{s,s'}$.

2.2.8.2 Mean Function

The mean function models the prior mean of the unknown function. The mean function is, however, a parametric function and the parameters of the mean function are called hyperparameters. Due to the lack of prior information, the behaviour of the mean function is either unknown or challenging to define. For example, future traffic speed is predicted using a GP. There is no prior information over the mean of the traffic speed. Typically, the mean function for traffic speed is a zero mean function [56]. In different contexts, it is useful to apply the non-zero mean function. For example, the mean function can be a constant or a linear function. The GP converges to the mean function in long-term prediction, and thus mean functions may play an effective role in application. Non-zero mean functions are listed as follows, **a) Constant:** The constant mean function is given below:

$$m_c(\mathbf{s}) = c, \tag{2.53}$$

where c represents a constant. The constant mean function and the zero-mean function have a similar behaviour, which means that the GP converges to the constant c in long-term prediction.

b) Linear: The linear mean function is given below:

$$m_{lin}(\mathbf{s}) = a\mathbf{s} + c, \tag{2.54}$$

where a is an unknown coefficient.

c) Basis: The basis mean function is given below:

$$m_{ba}(\mathbf{s}) = dg(\mathbf{s})^T, \tag{2.55}$$

where $g(\mathbf{s})$ is fixed basis function and d is a parameter of the mean function.

2.2.8.3 Covariance Function

the prior on the unknown function f can be implied by the covariance function. The covariance function contains the covariance information of the inputs. The same as mean functions, the covariance functions are parametric and determined by hyperparameters. The choice of the covariance function is often critical. Examples of commonly used covariance functions are given as follows:

a) Linear: The linear covariance function is given below:

$$K_{lin}(\mathbf{s}, \mathbf{s}') = \sigma_s^2(\mathbf{s} - c)(\mathbf{s}' - c) + \sigma_b^2, \tag{2.56}$$

where the magnitude and the bias variance are denoted as σ_v^2 and σ_b^2 respectively.

b) Exponential: The exponential covariance function is given below:

$$K_{exp}(\mathbf{s}, \mathbf{s}') = \sigma_v^2 \exp(-\frac{|\mathbf{s} - \mathbf{s}'|}{l}), \tag{2.57}$$

where σ_v^2 and l denote the magnitude variance and lengthscale hyperparameters. The width of the variations is controlled by the lengthscale, which means that the input is correlated over a long range when the lengthscale is large.

c) Squared Exponential: The squared exponential covariance function is given below:

$$K_{se}(\mathbf{s}, \mathbf{s}') = \sigma_v^2 \exp(-\frac{(\mathbf{s} - \mathbf{s}')^2}{2l}). \tag{2.58}$$

The squared exponential covariance function is the most commonly used kernel.

d) Rational Quadratic: The rational quadratic covariance function is given below:

$$K_{rq}(\mathbf{s}, \mathbf{s}') = \sigma_v^2 \exp(1 + \frac{(\mathbf{s} - \mathbf{s}')^2}{2\alpha l})^{-\alpha}, \tag{2.59}$$

where α is a scaling factor. The rational quadratic covariance function is identical to the squared exponential covariance function when $\alpha \to \infty$.

e) Periodic: The periodic covariance function is given below:

$$K_{per}(\mathbf{s},\mathbf{s}') = \sigma_s^2 \exp(-\frac{2\sin^2 \frac{|\mathbf{s}-\mathbf{s}'|}{p}}{l^2}), \tag{2.60}$$

where p implies the period. Defining $p = 2$ means that the period is 2π. The period covariance function models the periodic characteristics of the unknown function.

2.2.8.4 Regression Equations

the regression problem can be solved by Bayesian inference with a GP prior. Let $\mathbf{f}$ be the observed function values for the training set and let $\mathbf{f}_t$ be the set of function values corresponding to the test set $\mathbf{x}_t$. Therefore, the joint distribution is defined as follows:

$$\begin{bmatrix} \mathbf{f} \\ \\ \mathbf{f}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \\ \boldsymbol{\mu}_t \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_{s_t s} \\ \\ \mathbf{K}_{s_t s}^T & \mathbf{K}_{s_t s_t} \end{bmatrix} \right), \tag{2.61}$$

where $\boldsymbol{\mu}_t$ is the test mean, $\mathbf{K}_{s_t s}$ is the train-test covariance, and $\mathbf{K}_{s_t s_t}$ is the test-test covariance. Therefore, corresponding to the conditioning, the joint Gaussian prior distribution on the observations is given by

$$\mathbf{f}_t|\mathbf{S}_t, \mathbf{S}, \mathbf{f} \sim \mathcal{N}(\mathbf{K}_{s_t s}\mathbf{K}^{-1}\mathbf{f}, \mathbf{K}_{s_t s_t} - \mathbf{K}_{s_t s}\mathbf{K}^{-1}\mathbf{K}_{s_t s}^T). \tag{2.62}$$

Assuming additive independent identically distributed Gaussian noise with variance σ_n^2, the prior of the noisy observation becomes $cov(\mathbf{y}) = \mathbf{K} + \sigma_n^2\mathbf{I}$, where $\mathbf{y}$ are the ground truth observations. Following equation (2.63), the joint distribution of the observed values and the function values for test set can be written as,

$$\begin{bmatrix} \mathbf{y} \\ \\ \mathbf{f}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2\mathbf{I} & \mathbf{K}_{s_t s} \\ \\ \mathbf{K}_{s_t s}^T & \mathbf{K}_{s_t s_t} \end{bmatrix} \right). \tag{2.63}$$

The conditional distribution is, therefore, $f_t|\mathbf{S}, \mathbf{y}, \mathbf{S}_t \sim \mathcal{N}(\mathbf{K}_{s_t s}[\mathbf{K} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{f}, \mathbf{K}_{s_t s_t} - \mathbf{K}_{s_t s}[\mathbf{K} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{K}_{s_t s}^T)$. The predictive mean and covariance functions are typically parameterised in terms of hyperparameters θ. Subsequently, the training data and the hyperparameters are given as $\mathbf{s}, \mathbf{y} \in \mathcal{D}$ and θ, and therefore the mean and variance function of the predictive distribution $p(f_t|\mathcal{D}, \theta, \mathbf{s}_t) = \mathcal{N}(f_t|\mu(\mathbf{s}_t), \sigma_t^2(\mathbf{s}_t))$ at a test point $\mathbf{s}_t$ is formulated as follows:

$$\mathbb{E}(\mathbf{s}_t) = \mathbf{K}_{s_t s}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \tag{2.64}$$

$$\mathbb{C}(\mathbf{s}_t) = \mathbf{K}_{s_t s_t} - \mathbf{K}_{s_t s}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{K}_{s_t s}^T. \tag{2.65}$$

The variance in equation (2.65) is only based on inputs [56]. Variance is a difference between prior covariance, $\mathbf{K}_{s_t s_t}$, and the function information provided by the observations, $\mathbf{K}_{s_t s}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{K}_{s_t s}^T$. Therefore, the predictive distribution of the test target $\mathbf{y}_t$ can be simply calculated by adding $\sigma_n^2\mathbf{I}$ and the function variance $cov(\mathbf{f}_t)$.

2.2.8.5 Learning of Hyperparameters

the mean and covariance functions define a GP, while the mean and covariance functions depend on their hyperparameters. Although there are infinite combinations of the hyperparameters, they are selected based on the training data, which is called hyperparameter learning. Hyperparameter learning is performed by maximising the logarithmic marginal likelihood that is given below:

$$L = \log p(\mathbf{y}|\mathbf{v}, \theta) = -\frac{1}{2}|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi), \tag{2.66}$$

where $|\cdot|$ is the matrix determinant. With the hyperparameters obtained by maximising the log marginal likelihood, the prediction mean and the covariance matrix can be calculated using equation (2.64) and (2.65). Prediction variance can be extracted by selecting the diagonal elements in the covariance matrix. The GP methods provide predicted variances in addition to the predictions and therefore provide uncertainty information.

2.3 Introduction of Uncertainty Quantification

The uncertainty exist in different areas, from stock market price prediction to medical diagnosis. Nowadays, machine learning and deep learning models can investigate the uncertainties for statistical inference [58]. Before practical applications, it is critical to evaluate the efficacy of the model [59]. Uncertainty quantification (UQ) reinforces critical decisions since predictions cannot be trusted without UQ and may be inaccurate. The DL model is intended to achieve specific performance goals through training with labelled data. The iterative training process optimises the model parameters to provide satisfactory performance [60]. There are different uncertainties and common uncertainties are caused by the following reasons: 1) how well the collected data represent the actual phenomenon and distribution, 2) the accuracy and completeness of the training data, 3) the selection of the DL model, and 4) the performance of the models [61]. The predictions generated by the model are uncertain since they are prone to data noise, incorrect model inference, and inductive assumptions. Therefore, a trustworthy method is necessary to represent the uncertainty. A model that effectively handles uncertainty should provide better accuracy [60]. The uncertainty occurs due to the mismatch between the training and testing data or the presentation of irreducible data noise. Aleatoric uncertainty (data) and epistemic uncertainty (model) are the two major types of uncertainty [62].

Aleatoric uncertainty (AU) is irreducible uncertainty in the data that is the main reason causing uncertainty in the predictions, such as noise in the data. The AU is an inherent property of the data distribution. For example, in binary classification problem, the predictions located in two-class distribution intersection have higher AU compared to other data outside the intersection. Epistemic uncertainty (EU) is caused by insufficient knowledge of the data. The inadequate knowledge means that the number of data may be a large collection, but the collection of data may be uninformative [63]. In this case, the model is able to characterise the emergent features of the data. However, data collection is incomplete, noisy, multi-modal and discordant [58]. The distribution of model parameters formulates the EU [60]. In this section, two uncertainty quantification methods are introduced.

2.3.1 Confidence Interval

The GP is introduced in Section 2.2.8 as a powerful tool to establish models with additional uncertainty information. Since a GP models the distribution of an unknown function, the GP model outputs a mean function $m(x)$ and a covariance function $\mathbf{K}(x, x')$. Based on the prediction mean and variance, which is the diagonal element of the covariance matrix, a confidence interval can be constructed. A confidence interval provides the estimated interval where the ground truth will be located, and it only involves the prediction mean and variance [64]. Therefore, the following statement holds:

$$\mathcal{P}(|\mathbf{y} - \mu| < z\sigma) \geq \gamma, \tag{2.67}$$

where $\mathbf{y}$ is the ground truth, μ is the prediction mean, z is a constant, σ is the prediction standard deviation, and γ it the probability. In general, the confidence levels C are 0.99, 0.95 and 0.90, which correspond to 99%, 95% and 90% confidence that the interval covers the ground truth. Each confidence level C refers to a value of z, which is listed in the z-table [65]. For example, a 95% confidence interval is formulated as,

$$(\mu - z\sigma) < \mathbf{y} < (\mathbf{m} + z\sigma), \tag{2.68}$$

where z equals to 1.96 in case of 95% confidence level.

2.3.2 Variational Lower Bound

Let the input data be $\mathbf{X} \in \mathbb{R}^D$, output be $\mathbf{Y} \in \mathbb{R}^D$ and a set of latent variables $\mathbf{z}$. The Bayesian inference deals with the following posterior for the inferences:

$$p(\mathbf{z}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{y})}. \tag{2.69}$$

However, the denominator $p(\mathbf{y})$ is ether unavailable or computationally expensive [66]. Variational inference (VI) is designed to approximate this intractable posterior distribution [67]. In VI, a family $\mathbb{Q}$ of densities is specified for the latent variables, and each $q(\mathbf{z}) \in \mathbb{Q}$ is one of the candidate approximations of exact conditional in equation (2.69) [66]. The core idea of VI is to transform the approximation problem into an optimisation problem. The aim is to find the best candidate $q^*(\mathbf{z})$ that minimises the Kullback-Leibler (KL) divergence [68] to the true conditional as follows:

$$q^*(\mathbf{z}) = \mathbf{arg}_{q(\mathbf{z}) \in \mathbb{Q}} \min KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{y})), \tag{2.70}$$

which is a distance measurement between the two distributions. The KL divergence can be formulated as follows:

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}[\log q(\mathbf{z})] - \mathbb{E}[\log p(\mathbf{z}|\mathbf{y})] + \log p(\mathbf{y}) \tag{2.71}$$

However, the KL divergence still cannot be calculated, as $\log p(\mathbf{y})$ is intractable. Therefore, the Evidence Lower Bound (ELBO), an alternative objective equivalent to KL, is optimised:

$$ELBO = \mathbb{E}[\log p(\mathbf{z}|\mathbf{y})] - \mathbb{E}[\log q(\mathbf{z})]. \tag{2.72}$$

ELBO is indeed the negative of equation (2.71) added with $\log p(\mathbf{y})$. Then, ELBO can be represented as $ELBO = \log p(\mathbf{y}) - KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}))$, which is a constant with respect to $q(\mathbf{z})$ [66]. Therefore, minimisation of KL divergence becomes maximisation of ELBO.

For a GP, as described in the previous sections, a GP is $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ and the latent variable $\mathbf{z}$ is sampled from a Gaussian distribution. The output y is then distributed based on the likelihood function $y|f \sim h(f)$. Inducing samples are used, and therefore, the variational lower bound is optimized as

$$\log p(\mathbf{y}) \geq \mathbb{E}[\log p(\mathbf{y}|f)] - KL(q(f_z)||p(f_z)), \tag{2.73}$$

where $\mathbf{z}$ is the latent variable or inducing points.

2.4 Fast Edge Detection Using Structured Forests for Classification and Segmentation

In this section, traditional edge detection for segmentation is introduced.

Since the 1970s, edge detection has become a basic element for computer vision [69, 70, 71]. Edge detection is a critical process in various computer vision applications, especially image segmentation [72, 73]. Traditional edge detection methods calculate color gradients using a variety of methods, such as non-maximum suppression [74], where non-maximum suppression is an algorithm to select one entity out of many overlapping entities. However, a number of visually salient edges correspond to brightness, textures, or illusory contours [75] rather than colour gradients. Brightness, color, texture and depth gradients are taken as input to the state-of-the-art edge detection algorithm [73, 76, 77].

Edges within a local patch are highly interdependent [78]. Edges include patterns, for example, lines and junctions [79]. Problems with similar characteristics are solved using structured learning [80]. To obtain the advantage of the inherent structure of the edge and remain computationally efficient, structured learning is applied to edge detection [81]. Structured learning focuses on the challenge of defining a mapping with an arbitrarily complex input or output space [81, 82, 83, 84]. Structured random forest is introduced in the following sections.

The structured random forest is inspired by the work of Kontschieder et al. [85]. The key observation of this work is that the leaf node can store any type of output and is independent of the structured labels with a certain image patch [81]. Based on this observation, structured random forest was proposed and has been applied to a wide range of output spaces.

2.4.1 Random Decision Forests

Beginning with the introduction of random decision forests [86, 87], the input $\mathbf{x}$ is classified by iteratively splitting through a decision tree. Each node is composed of a binary split function,

$$h(x, \theta_j) \in \{0, 1\}, \tag{2.74}$$

where parameter required to be optimised is denoted as θ_j at branch j, and the input is represented as x. If $h(x, \theta_j) = 1$ the node is split to the right, otherwise left, with the process terminated at the leaf node. For a given input $\mathbf{x}$, the output, stored at the leaf node, of the decision tree can be either a deterministic label or a distribution.

A collection of independent decision trees f_t forms a decision forest. The ensemble model combines the predictions $f_t(\mathbf{x})$ into one prediction output. The choice of the ensemble model depends on the target labels $\mathcal{Y}$. Majority voting is the common choice for classification and averaging is the common for regression. The leaf node of a decision tree may store arbitrary information [81]. Furthermore, the leaf node only relies on the input, and therefore Kontschieder et al. [85] suggested that complex output space, such as structured outputs, can be used.

Although prediction of decision forests is straightforward, training is more challenging. The review of standard learning procedure is demonstrated next, and the description of learning with structured output is introduced.

2.4.1.1 Training Decision Trees

Each decision tree is independently trained. Given a nod j and a training set D_j, the training procedure aims to chose a value of θ_j of a split function $h(x, \theta_j)$ to maximise of the information gain I at each node:

$$I_j = I\left(D_j, D_j^L, D_j^R\right), \tag{2.75}$$

where, $S_j \in \mathcal{X} \times \mathcal{Y}$ is the training set with $\mathcal{X}$ samples and $\mathcal{Y}$ labels. $S_j^L = \{(x, y) \in S_j | h(x, \theta_j) = 0\}$ represents proceeding on the left node and $D_j^R = D_j / D_j^L$ represents the proceeding at the right node, where $x \in \mathcal{X}$ is one of the samples. The training procedure stops when it reaches the maximum depth or when the information gain drops below a fixed threshold.

The information gain for multiclass classification is then defined as:

$$I_j = H(D_j) - \sum_{k \in L,R} \frac{|D_j^k|}{|D_j|} H(D_j^k), \tag{2.76}$$

where $H(S)$ represents the Shannon entropy. Continuous forms of entropy and information gain can be extended to solve regression problems [88]. Minimising the variance of the labels at the leaf nodes is a common approach for single-variate regression [89]. With a mild additional assumption about $\mathcal{Y}$, a more general information gain criterion can be defined. Before going into the details about the structured random forest, the role of randomness played in the training procedure is discussed next.

2.4.1.2 Randomness and Optimality

A single decision tree has high variance and tends to overfit [89, 86, 87, 90]. Decision forests solve this problem by combining multiple independent and well trained trees. Therefore, achieving sufficient diversity of trees is the most critical component of the training procedure.

The diversity of the decision trees is achieved by randomly subsampling the training data or the features and splits in each node [90]. Node-level randomness tends to provide higher precision [87]. Specifically, only a limited set of parameters θ_j is sampled and tested, while optimising the information gain criterion shown in Eq. 2.75.

As a result, to obtain a high diversity ensemble, the accuracy of the individual decision tree is sacrificed [87]. Taking advantage of a similar intuition, an approximate criterion of information gain for structured labels is described below, which leads to the general structured forest [81].

2.4.2 Structured Random Forests

The structured random forests are extended from the random decision forests with a more general structured output space $\mathcal{Y}$. In the certain case of computer vision, the input is an image $x \in \mathcal{X}$, and the output is the corresponding annotations, such as segmentation masks or semantic labels. Random forests with structured labels face two challenges during the training procedure. The first is the high dimensionality and complexity of the output spaces. Therefore, evaluating a large number of candidate splits is extremely expensive [81]. Second, the information gain of the structured label is not properly defined.

Dollar et al. [81] suggest that the approximate information gain is sufficient to train the classifier [87, 85]. The philosophy is mapping the structured labels into a set of discrete labels $c \in \mathcal{C}$, where $\mathcal{C}$ is a set of discrete integers. Therefore, structured labels that have similar properties can be classified into the same category c. The measurement of similarity is highly demanded to calculate the information gain. However, the formulation of similarity over $\mathcal{Y}$ is not properly defined for some structured output space, particularly for edge detection. Instead of directly calculating similarity, an intermediate space $\mathcal{Z}$ is introduced to easily measure the distance. The two-stage approach is, therefore, utilised: first mapping $\mathcal{Y} \rightarrow \mathcal{Z}$ and then mapping $\mathcal{Z} \rightarrow \mathcal{C}$.

2.4.2.1 Intermediate Mapping

The core assumption is that a mapping can be defined in the following,

$$\Pi : \mathcal{Y} \rightarrow \mathcal{Z}. \tag{2.77}$$

The Euclidean distance over $\mathcal{Z}$ is used to approximate the dissimilarity of the labels. A binary vector $\mathbf{z} = \prod(y)$ can be defined to encode the class information of every pair of pixels. The intermediate space $\mathcal{Z}$ makes it easier to calculate the distance. However, $\mathcal{Z}$ can be high-dimensional, and it results in a challenge in computational resources. In the edge detection problem, there may be a large number of unique pixel pairs, and therefore, it is computationally expensive to calculate z for every y. Therefore, an approximate distance measure is applied to reduce the dimensinality of $\mathbf{z}$ [81].

For reducing the size of $\mathbf{z}$, m dimensions of $\mathbf{z}$ are sampled, which causes a reduced mapping. Different mappings are generated randomly and applied to the labels y during the training procedure. Therefore, with this kind of mapping, computational efficiency is improved and extra randomness is involved, ensuring the diversity of the decision trees. In the different aspects, Principal Component Analysis (PCA) [91] is another way to further reduce the dimension of $\mathcal{Z}$. PCA removes the noise in $\mathcal{Z}$ and approximately preserves the Euclidean distance [81].

2.4.2.2 Information Gain Criterion

There are a number of possible choices of the information gain criterion when the mapping is given as $\prod_{\phi} : \mathcal{Y} \rightarrow \mathcal{Z}$. Multivariate joint entropy can be computed directly for the intermediate space [81]. Due to the possible high complexity ($\mathcal{O}(|\mathcal{Z}^m|)$, where m can be large), the information gain can be determined by variance or a continuous formulation of entropy [88], with a given $\mathcal{Z}$.

Two ways of mapping structured labels into discrete labels are considered. The first is to cluster $\mathbf{z}$ into k classes with K-means. The second is to use PCA quantisation. These two approaches have similar performance, but the latter is faster [81].

2.4.2.3 Ensemble Model

Finally, the way to combine all labels into one prediction output is defined. As introduced in **b)**, a m dimensional mapping is sampled and $\mathbf{z}$ is computed. The label y_k with a minimum overall distance of z_k is selected. Since any predicted labels must be observed in the training procedure, the ensemble model cannot recognise new labels [81]. Therefore, domain-specific ensemble models are preferred in practise.

2.4.3 Edge Detection

This section describes how structured forests work for edge detection. The input of the method is a multiple-channel image. Similarly to the semantic problem, the aim is to classify each pixel with a binary label, which indicates the property of the pixel. It is assumed that a segmented training set is given, and the boundaries between the segments corresponding to the contours are annotated for each patch [72, 92]. For a given patch, the annotation can be either a segmentation mask or a binary edge map. As denoted above, $y \in \mathcal{Y} = \mathcal{Z}^{d \times d}$ is defined as the segmentation mask, and $y' \in \mathcal{Y}' = \{0, 1\}^{d \times d}$ is defined for the binary edge map, where d represents the patch size. An edge map can be obtained from a segmentation mask and, therefore, both annotations are utilised [81]. How to calculate the input features x, the mapping function to determine the splits, and the ensemble model is described below.

2.4.3.1 Input features

Suppose the model aims to provide a 16×16 segmentation mask for a 32×32 image. The model first enhances the image with additional K channels and thus generates a vector of features $x \in \mathcal{R}^{32 \times 32 \times K}$. Two features are used, named pairwise difference and pixel lookups.

Following suggestions provided by Lim et al. [78], color and gradient channels are used. The image is augmented into 13 channels, and details can be found in [81]. The channels are blurred by a triangle filter with a radius of 2 pixels and downsampled by a factor of 2, and hence, there are 3328 candidate features x. Also, inspired by [78], the pairwise difference features are computed. The channels are blurred by a large triangle filter with a radius of 8 pixels and downsampled to a size of 5×5. An additional 300 candidate features for one channel are generated by sampling candidate pairs and computing the difference, and therefore, there are 7228 total candidate features [81].

2.4.3.2 Mapping function

In the training procedure of the decision trees, a mapping $\prod : \mathcal{Y} \rightarrow \mathcal{Z}$ is required to be defined. One choice is to map $\mathcal{Y}$ to the binary edge map. However, the Euclidean distance used to measure the distance is brittle over the binary edge map. Therefore, an alternative mapping is defined. Let $y(j)$, where j is between 1 and 256, represent the segment index on the j-th pixel. Two not identical pixels can be sampled and checked if they are in the same segment, which defines a long binary vector encoding $[y(j_1) = y(j_2)]$ for each unique pair of pixels.

2.4.3.3 Ensemble model

Combining the output of multiple trees contributes to the robust results of random forests. By averaging the multiple overlapped edge maps, a soft edge can be generated, as merging the segmentation masks for the overlapping patches is difficult [81]. Thus, the edge map at each leaf node is stored for learning the mask, which allows predictions to be combined.

2.4.3.4 Multi-scale Detection (SE+MS)

Two enhancements are introduced in this section. A multi-scale version (MS) of the edge detector is implemented, and the idea is inspired by Ren [93]. The results generated by the structured edge (SE) detector are averaged after being resized to the original dimension. This approach critically increases edge quality.

2.4.3.5 Edge Sharpening (SE+SH)

It is observed that the structured edge detector effectively detects edge peaks of isolated edges. However, the edges of fine image structures blur together resulting in miss detection, and no clear peak emerges for weak edges. The diffuse edge is mainly caused by the noisy edge map predictions, and the predictions cannot be effectively assigned to each other. Specifically, the overlapping predictions of the edge shift several pixels from the ground truth location.

To avoid this phenomenon, the sharpening procedure is introduced to align the response of the edges from overlapping positions. The image color and depth are useful for more precise localisation of the predictions. The predicted segmentation mask can be transformed implicitly, and thus the mask better matches the original image [81]. Matching overlapping masks to the original image connects each mask, which generates better localised edges.

The better aligned masks are produced by sharpening the predicted segmentation mask and the corresponding original image. The mean colour $\mu_s = \mathbb{E}[x(j)|y(j) = s]$, where j is the pixel index, is computed for each segment s. Then, the assigned segment for each pixel is updated by assigning it to the segment that minimises $||\mu_s - x(j)||_2$ [81]. The assignable segments are restricted to the immediately adjacent segments (4-connected neighbourhoods). With the sharpened segmentation masks, the corresponding edge maps are computed and averaged. The aggregated edge map is sharper, since the edge maps are assigned to the corresponding images in a proper way. Edge sharpening may be performed several times before averaging the edge maps. Since the edges are sparse, sharpening can be implemented efficiently.

2.5 Summary

Chapter presents an overview of the main widely used methods for traffic prediction and, respectively, for edge-based object detection in images. Section 2.1 provides a brief introduction to machine learning methods. The main existing methods for traffic prediction are reviewed in Section 2.2. Starting with a model-based method, Section 2.2 first introduces SRK, which brings a new perspective on capturing spatial features and spatial homogeneity of road segments. However, the physical properties of the road segments are required for developing the SRK, and thus different SRK models need to be developed with different traffic networks, which is either expensive or time-consuming.

Therefore, the data-driven methods are demonstrated. The ARIMA is a popular prediction model and achieves success in short-term highway traffic prediction. However, the model mainly considers the evolution of traffic over time. In addition, the ARIMA is a linear model, and thus only linear relationships between data can be captured, which means that ARIMA is not able to capture complex patterns within the data. The ANNs are then widely applied to traffic predictions because of their generalisability and the capability of handling multi-dimensional data. The BCNN is proposed to combine multiple outputs generated by ANNs. With multiple predictors, BCNNs have a stronger ability to capture different complex patterns within traffic data. The LSTM-NN is introduced next as it learns the long-term dependency. Traffic data within a traffic network are correlated not only over time but also over space. Therefore, CNN, a deep learning method, is introduced to abstract the spatio-temporal features from traffic data which are converted into the form of an image. The CNN framework, proposed by Ma et la. [45] will be the base-line methods for the traffic prediction in this thesis. Besides, the structural RNN is introduced for traffic prediction, which provides another perspective of utilizing the spatio-temporal features of the traffic speed data. Parametric models have achieved great success, as introduced in previous sections. On the other hand, non-parametric models, including LS-SVM and GP, provide a consideration of uncertainty, especially the GP models. The GP has the potential to equip the DL methods with uncertainty quantification. The next two sections focus on the traffic prediction problems. Model-based methods are briefly introduced for traffic prediction. However, the dynamics of the traffic systems is required to be explicitly defined for model-based methods, and thus, data-driven models are described to overcome the disadvantages of model-based methods. ARIMA, BCNN, LSTM-NN are introduced as the statistical and machine learning methods for traffic prediction, and they provide reliable prediction, but the spatio-temporal information within the traffic data is not concerned. Therefore, the CNN, a deep learning method, is introduced to abstract the spatio-temporal features from traffic data which is converted into the form of an image. The CNN framework, proposed by Ma et la. [45] will be the base-line methods for the traffic prediction in this thesis. Besides, the structural RNN is introduced for traffic prediction, which provides another perspective of utilizing the spatio-temporal features of the traffic speed data. Parametric models have achieved great success, as introduced in previous sections. On the other hand, non-parametric models, including LS-SVM and GP, provide a consideration of uncertainty, especially the GP models. The GP models will be the focus, and a GP related framework will be proposed in the chapter 3. The uncertainty attracts the interest of researching, not only because uncertainty quantification helps to improve the accuracy of the models, but also quantifies how much the predictions are trustworthy. Section 2.3 introduces two types of uncertainty and two methods for uncertainty quantification. The novel approaches for traffic speed prediction proposed in Chapter 3 are based on the following:

- The traffic speed changes are represented as an image. First, the measured traffic speed data are converted into an image (matrix) representation as described in Section 2.2.5.1.
- The CNN has the ability to extract features, but uncertainty characterisation is not provided. Since the GP is a powerful Bayesian framework, it is adopted to characterise the results, and this forms the CNN-GP framework. Next, two novel algorithms are proposed, one is the ConvNet GP and the CNN-GP, and they are shown to achieve accurate traffic speed prediction and in addition provide uncertainty characterisation based on the calculated GP variance.
- A confidence interval is applied to analyse the uncertainty.

Section 2.4 introduces traditional methods for image segmentation, and fast edge detection using structured forests is described in detail, as it will be used as the base-line methods for evaluating the machine learning approaches proposed in this thesis. Fast edge detection using structured forests is the base-line of this case study. In Chapter 4, novel approaches for cancer bone segmentation is based on the following:

- The fast edge detection using structured forests described in Section 2.4 is the base-line of this case study.

- The approaches developed for traffic speed prediction are modified to the segmentation task, providing a novel perspective on the segmentation of cancer bones.
- The deep learning approach is developed to achieve segmentation of the lesion area autonomously.

The next chapter proposes machine learning frameworks for traffic speed prediction and an approach for uncertainty quantification named the uniform error bound approach. A detailed performance validation is provided for the proposed approach.

Chapter 3

Machine Learning for Traffic Prediction

3.1 Introduction

The previous chapter gives an overview of traffic prediction methods. This chapter proposes a capsule network and two GP related approaches for short-term traffic prediction. Two long-term traffic prediction frameworks based on the GP are described and the uniform error bound is applied for uncertainty quantification. All proposed approaches are evaluated with real traffic speed data collected in Santander, Spain. The uniform error bound is first evaluated with simulated data and then with real traffic speed data.

Section 3.2 proposes a capsule network and two GP related approaches for short-term traffic speed prediction, and the novelty and significance are the following:

- A efficient architecture of CapsNet is proposed for traffic speed prediction, which is the first time applied on the traffic prediction problem. Compared to CNN, the proposed CapsNet can extract higher-level features and encode the probability of those features that are located in the local region.

- A ConvNet GP is proposed for traffic speed prediction by equating the base-line CNN to a GP, which provides similar performance as the base-line CNN and additional uncertainty information.

Section 3.3 proposes a novel deep kernel CNN-GP framework with spatio-temporal kernels. The contributions are the following:

- CNN-GP framework for time series prediction is proposed. CNN provides feature maps to the GP regression which makes the prediction and quantifies the impact of uncertainties.
- A multitype spatio-temporal GP is proposed to better encode the prior knowledge observed from the data
- A detailed performance validation and evaluation of these proposed approaches is conducted on two case studies: (1) Seasonal carbon dioxide (CO_2) prediction, (2) Traffic prediction in terms of both volume and speed data.
- The proposed approach with spatio-temporal kernels can provide both short-term and long-term predictions.

Section 3.4 proposes a modified uniform error bound (UEB) to quantify the uncertainty of the traffic speed prediction problem, and the novelty and contribution are the following:

- A effective UEB, with weaker assumptions comparing with other error bound methods, is proposed for CNN-GP framework, which provides uncertainty analyses for traffic speed predictions.
- The detailed performance validation and evaluation of UEB are conducted on simulated and real data.

3.2 Machine Learning Frameworks for Short-term Traffic Prediction

In this section, deep learning and a GP framework are introduced. CapsNet is first introduced, and a novel architecture of CapsNet is proposed that encodes the prior information of the traffic features, which is the first time applied to the traffic prediction problem. Second, the ConvNet GP is described, and it is a way to enhance the CNN with the ability to analyse uncertainty information.

3.2.1 Capsule Network

As Section 2.2.5 introduced, CNN are powerful to comprehend images due to its convolution and pooling properties. Geoffrey Hinton emphasised his confidence in convolution, and, however, he presented four primary arguments for pooling as a routing method in his talk [94]. The arguments against pooling are introduced as following. **Pooling is unnatural.** Hinton stated that pooling badly fits the psychology of shape

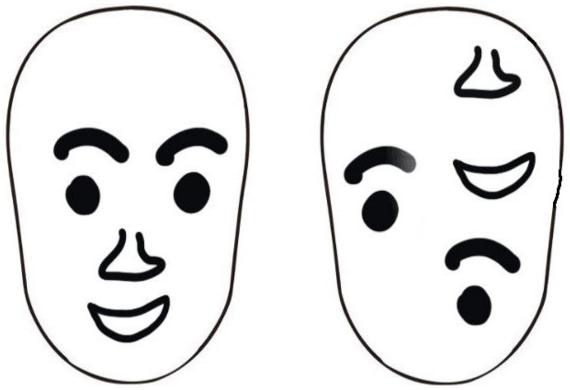


Figure 3.1: *CNNs classified both images as human face.*

perception [94]. Human detects an object instantaneously as it appears. Based on the information of the object observed, a person directs the information to the part of the brain that can process it best. On the other hand, the pooling operation distributes the most active information to all subsequent neurons [95], which is out of the ordinary and inhibits the network from learning about the detailed information. **Pooling Solves a wrong problem.** By pooling the neural activities together, CNNs attempt to make the neural activity invariant to small pose changes [94]. This brings one advantage for classification tasks, since the label should remain consistent regardless of the locations of the objects (spacial invariant). Fig. 3.1 shows that CNNs can only abstract and detect features, but cannot understand the relationship between features, which means that both images are accurate portraits of a human face. **Pooling does not use the linear structure of vision.** The transformation, such as rotation and wrap, changes a sizable portion of the pixels in the image. Therefore, the viewpoint is the primary cause of the image variance [94]. It is a linear problem for a person to compose a scene of multiple components, since the person can recognise the relationships between the components. Without this linear framework, object detection in computer vision is no longer linear, but more complex. As a result, a typical CNN must be exponentially increased in parameters size [96] and trained with an exponentially growing amount of data, such as applying data augmentation [97]. **Pooling is a poor way to perform dynamic routing.** The most prominent activations are gathered by pooling and propagated to the same neurons in the following layer. However, an entirely different collection of neurons is responsible for managing different types of activations, as input images are translated. Human vision can understand the translation of the image and activates the same neurons, which occurs at runtime, and hence it is dynamic rather than statically preconnected [95]. This is a routing problem that the features of the image pixels must be correctly routed to the neurons that specialise on that type of feature [94]. Dynamic routing selects the best neuron, while pooling selects the best inputs.

To conclude the limitations of CNNs brought, the major challenge is that the CNNs are not able to recognise pose, texture and transformation [96], which is caused by the invariance brought by the pooling. Therefore, the CNNs lack equivalence. On the other hand, the pooling operation loses features in the input [98]. As the results show, a significant number of training data are required to compensate for the loss. For achieving the best results, CNNs are deep in depth, and hence a large amount of parameters is calculated [99]. CNNs are more vulnerable to adversarial attacks and generate wrong results [100].

Hinton et al. [101] proposed the concept of capsule networks to emphasise the ability of a network to recognise poses. Hinton defined capsules as groups of neurons divided into each layer [96]. Sabour et al. [96] further improved the capsule networks. Fig. 3.2

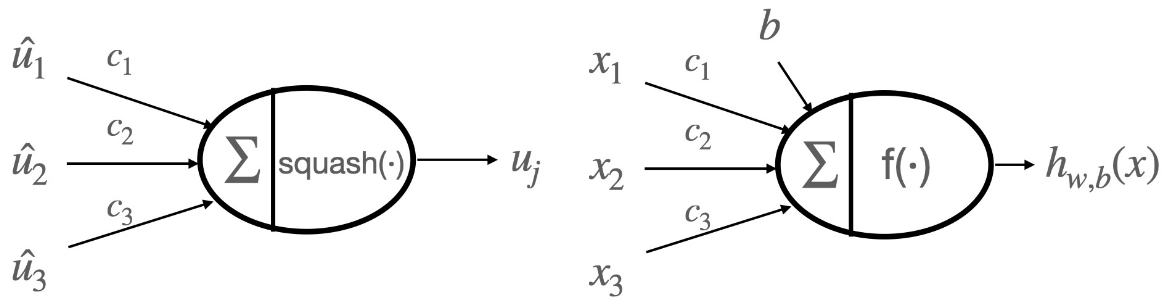


Figure 3.2: *Left shows the capsule, and right represents the neuron.*

presents a comparison between a capsule and an artificial neuron. A neuron calculates a scalar output from a list of scalar inputs, while a capsule computes a vector output from a list of vector inputs by wrapping a group of neurons. In this way, a capsule encodes instantiation parameters such as position and pose, and the length of the output vector represents the probability of the existence of the feature [96].

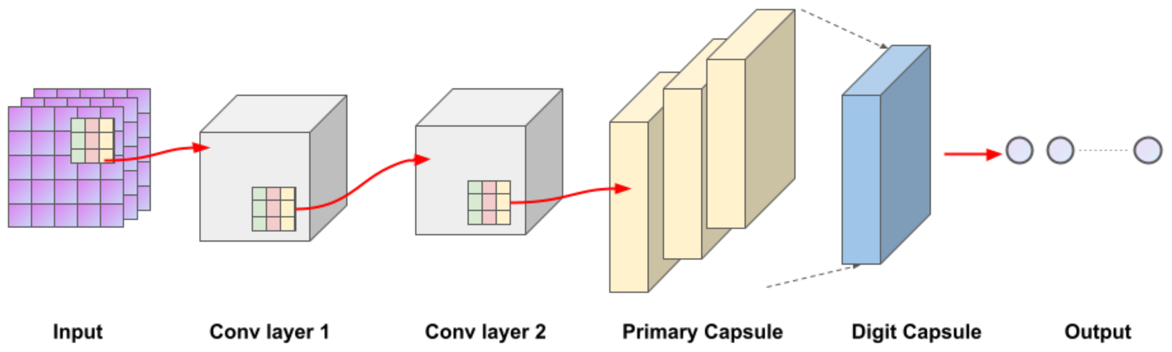


Figure 3.3: *The architecture applied on traffic speed prediction for complex traffic network.*

A capsule network consists of convolutional layer, primary capsule layer and class capsule layer. The convolutional layers are used to extract the features from the data and the outputs of the convolutional layers are fed into the primary capsule layer. The activity vector for each capsule i in layer l is denoted as $\mathbf{u}_i$, and the output vector $\mathbf{u}_i$ of i-th capsule in layer l is then fed into all the capsules in the next layer $l+1$ [98]. The j-th capsule in layer $l+1$ produces u_i with the corresponding weight $\mathbf{W}_{i,j}$, and the output vector of the capsule in layer $l+1$ is $\hat{\mathbf{u}}_{j|i}$. The transformation of the predicted vector is represented as

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{i,j}\mathbf{u}_i. \tag{3.1}$$

The predicted vector $\hat{\mathbf{u}}_{j|i}$ and the coupling coefficient $\mathbf{Cop}_{i,j}$ are multiplied to obtain a single primary capsule prediction. A weighted sum $\mathbf{sum}_j$ is applied to all primary capsule predictions, and the coupling coefficient between the capsule i and other capsules is determined by **softmax**,

$$\mathbf{sum}_j = \sum_{i=1} \mathbf{Cop}_{i,j}\hat{\mathbf{u}}_{j|i}, \qquad \mathbf{Cop}_{i,j} = \frac{exp(b_{i,j})}{\sum_k exp(b_{i,k})}. \tag{3.2}$$

The outputs are fed into the squashing function $\mathbf{sum}_j$,

$$\mathbf{squash}_j = \frac{\|\mathbf{sum}_j\|^2\mathbf{sum}_j}{1+\|\mathbf{sum}_j\|^2\|\mathbf{sum}_j\|}. \tag{3.3}$$

The non-linear squash function shrinks the length of the output of the capsule between 0 and 1. The output of the capsule layer $\mathbf{sum}_j$ is routed to the next layer capsule. The coupling coefficient $\mathbf{C}_{i,j}$ ensures that the prediction of capsule i in layer l is correctly routed to that of capsule j in layer $l+1$. The output vector corresponding to each capsule can be divided into two parts: the probability that represents the existence of the feature and the set of instantiation parameters [98]. Therefore, when lower-level capsules agree on a higher-level capsule, the construction of relationships between capsules in different layers is called dynamic routing-by-agreement [96]. In contract to the pooling, routing algorithm is applied at runtime, and the goal is to redirect previous capsule output vectors to the following capsules where they agree with their input [95].

Algorithm 2 Routing Algorithm [96]

```
procedure: $Routing(\widehat{u}_{j|i}, r, l)$
  for all capsules $i$ in layer $j$ and capsule $j$ in layer($l$+1): $b_{ij} \leftarrow 0$
  for r iteratoins do
    for all capsules $i$ in layer $l$: $c_i \leftarrow softmax(b_i)$ Eq.3.2
    for all capsules $j$ in layer ($l$+1): $x_i \leftarrow \sum_i c_{ij}\widehat{u}_{j|i}$
    for all capsules j in layer ($l$+1): $v_j \leftarrow squash(x_j)$ Eq.3.3
    for all capsule $i$ in layer $l$ and capsule $j$ in layer (l+1): $b_{ij} \leftarrow b_{ij} + \widehat{u}_{j|i}v_kj$
return $v_j$
  end for
```

The architecture proposed for the traffic speed prediction for complex road network is presented in Fig. 3.3. The architecture begins with two convolutional layers that abstract simple features from the spatio-temporal traffic speed data in the form of images. Each convolutional layer has a kernel size of 3×3 and 32 channels with a stride of 1 with zero padding. The third layer is the primary capsule layer that extracts higher-level features and encodes the spatial relationships between features. Therefore, the outputs of the primary capsule are vectors that contain high-level features and the probability of those features that are located in the local region. The capsules in the primary capsule layer output 8-dimensional vectors and the capsules in a cuboid share weights. The output of the primary capsule is squashed to a fixed length and fed into the digit capsule. The last layer is the digital capsule that is used to generate traffic speed predictions and has a 16-dimensional capsule for each road segment. The dynamic routing algorithm is performed between the primary capsule and the digit capsule, which determines the connections between the capsules in the primary capsule and the digital capsule. The association between each capsule representing a road segment and the other capsules in the primary capsule layer is captured by the dynamic routing algorithm. In this way, the digit capsule layer's capsules can be characterised by any remote local attribute.

Table 3.1: *Layer parameters of CapsNet*

Layer	Parameter	Activation
Convolution1	(32,3,3)	ReLu
Convolution2	(32,3,3)	ReLu
PrimaryCaps	(128,3,3) Capsule size 8	Relu
DigitCaps	Capsule size 16	-

Although CapsNet overcomes the drawbacks of CNN and encodes prior information with features, both CNN and CapsNet are deterministic models that lack uncertainty analyses. The next section demonstrates a framework that combines CNN and GP, which is one of the proposed approaches to equip CNN with uncertainty information by using GP.

3.2.2 Deep Convolutional Neural Network as Shallow Gaussian Process

A deep neural network can conceptually be considered as a Gaussian process regression [102]. Therefore, this vision is adopted and a deep CNN can be considered as a shallow GP [6]. As pointed out in [103] combining the non-parametric nature of GP regression and the learning ability of neural networks can improve the generalisation capabilities of the GPs.

3.2.2.1 Concepts and Definitions

As mentioned previously, inaccurate uncertainty estimation of CNNs is becoming increasingly problematic. For a standard CNN, with L hidden layers, the transformation from layer l to layer $l+1$ is given as following:

$$\mathbf{a}_j^{(l+1)}(\mathbf{S}) = \mathbf{b}_j^l + \sum_{i=1}^{C^l} \mathbf{W}_{j,i}^l \phi(\mathbf{a}_i^l(\mathbf{S})), \tag{3.4}$$

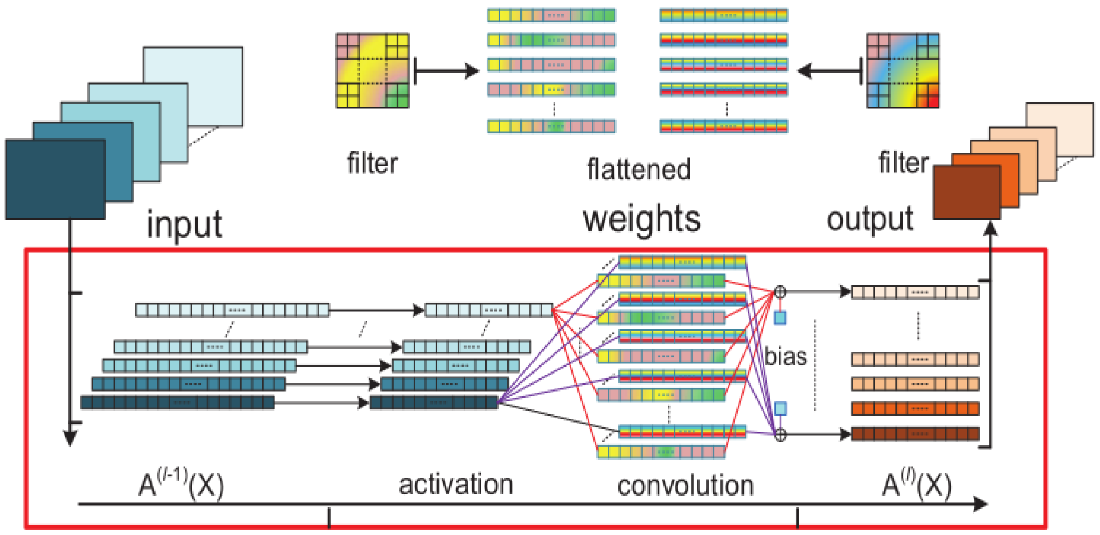


Figure 3.4: *Elementwise transformation of a CNN [6].*

where

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ . \\ . \\ . \\ \mathbf{s}_{C^0} \end{bmatrix}, \tag{3.5}$$

is the input image with height H^0 and width D^0. Each input image has C^0 channels, and, therefore, the input image is considered as a $C^0 \times (H^0D^0)$ matrix, with $\mathbf{s}_i$ a row vector of size $1 \times (H^0D^0)$. The i-th output from the l-th layer is represented by $\mathbf{a}_i^{l+1}(\mathbf{S})$. The bias is $\mathbf{b}_j^l$ and the weight matrix derived from the filter $\mathbf{U}_{j,i}^l$ on the l-th layer is $\mathbf{W}_{j,i}^l$. The activation for the output of the previous layer is represented as $\phi(\mathbf{a}_i^l(\mathbf{S}))$. For the first layer, $\phi(\cdot)$ simply maps the input $\mathbf{s}_i$ to itself. In equation (3.4), $\mathbf{W}_{j,i}^l\phi(\mathbf{a}_i^l(\mathbf{S}))$ indicates the dot product of $\mathbf{W}_{j,i}^l$ and $\phi(\mathbf{a}_i^l(\mathbf{S}))$.

In the traditional CNN paradigm, the elements of a convolution filter $\mathbf{U}^l_{j,i}$ are determined. One filter is usually responsible for a specific feature. It is hard to say if the filter can still manage to capture the features when the data is polluted by random noise. To tackle the problem, one intuitive idea is to make the filter itself random rather than determined. In this way, for one specific convolution filter $\mathbf{U}^l_{j,i}$, when the elements change randomly, the number of potential filters could approach infinity. Therefore, all filters together should be able to average the noise and extract the features from the polluted data as described in [6]. A convolution filter $\mathbf{U}^l_{j,i}$ is substantially a matrix as shown in Fig. 3.4. As in [6], each element $\mathbf{u}^l_{j,i,x,y}$ of $\mathbf{U}^l_{j,i}$ is governed by a Gaussian distribution as in equation (3.6), and the bias $\mathbf{b}^l_j$ is governed by another Gaussian distribution in the form

$$u^l_{j,i,x,y} \sim \mathcal{N}(0, \frac{\sigma^2_w}{C^l}), \tag{3.6}$$

where i and j are the channel index in layer $l-1$ and l, x and y are the horizontal and vertical location of the element in the filter

$$b^l_j \sim \mathcal{N}(0, \sigma^2_b). \tag{3.7}$$

As each single element $\mathbf{u}^l_{j,i,x,y}$ of $\mathbf{U}^l_{j,i}$ is subject to a Gaussian distribution, as many filters as possible can therefore be derived by sampling from the corresponding distribution. Each filter can be further flattened into a weight matrix $\mathbf{W}^l_{j,i}$ with the dimension of $N^l \times (H^l D^l)$. According to equation (3.4), if $N^l \to \infty$, then $C^l \to \infty$. With the Central Limit Theorem (CLT), equation (3.6) and (3.7), $\mathbf{a}^{l+1}_j(\mathbf{S})$ subject to a Gaussian distribution as $C^l \to \infty$.

3.2.2.2 Mean and Covariance

In [6], the authors show that, with certain constraints, a deep convolutional neural network is equivalent to a shallow Gaussian process. When it comes to a Gaussian process, the question of how to obtain the mean and covariance from a deep convolutional neural network is required to be investigated.

According to equation (3.4), the input is an image $\mathbf{S}$. To derive the covariance, at least another input image denoted as $\mathbf{S}^{'}$ is required. Indexed by $\mathbf{S}$ and $\mathbf{S}^{'}$, two feature maps $\mathbf{a}_j^l(\mathbf{S})$ and $\mathbf{a}_j^l(\mathbf{S}^{'})$ can be concatenated as

$$\mathbf{a}_j^l(\mathbf{S}, \mathbf{S}^{'}) = (\mathbf{a}_j^l(\mathbf{S}), \mathbf{a}_j^l(\mathbf{S}^{'}))^T, \tag{3.8}$$

which can be further extended as equation (3.9) to show the transformation from layer l to layer $l+1$,

$$\mathbf{a}_j^{l+1}(\mathbf{S}, \mathbf{S}^{'}) = \mathbf{b}_j^l \mathbf{I} + \sum_{i=1}^{C^l} \begin{bmatrix} \mathbf{W}_{j,i}^l & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{j,i}^l \end{bmatrix} \phi(\mathbf{a}_i^l(\mathbf{S}, \mathbf{S}^{'})), \tag{3.9}$$

which is proven to be multivariate Gaussian in [6] when $C^l \to \infty$. In equation (3.9), $\mathbf{I}$ is the identity matrix. The attribute holds for any given feature map determined by j and l, which are further constrained by the structure of the deep convolutional neural network. For more details, refer to [6].

To derive the mean and convariance, the element-wise feature map is first given as

$$\mathbf{A}_{j,g}^{l+1}(\mathbf{S}) = b_j^l + \sum_{i=1}^{C^l} \sum_{h=1}^{H^l D^l} \mathbf{W}_{j,i,g,h}^l \phi(\mathbf{A}_{i,h}^l(\mathbf{S})), \tag{3.10}$$

where l and $l+1$ are the indexes of the hidden layers, i and j are the indexes of the input and output channels, and h and g denote the location of the element within the input and output channels. When the input image is $\mathbf{S}^{'}$, the element-wise feature map becomes $\mathbf{A}_{j,g}^{l+1}(\mathbf{S}^{'})$.

With equation (3.10), the mean and covariance are formulated as in equation (3.11) and (3.12), respectively.

$$\mathbb{E}[\mathbf{A}^{l+1}_{j,g}(\mathbf{S})] = \mathbb{E}[\mathbf{b}^l_j] + \sum_{i=1}^{C^l} \sum_{h=1}^{H^l D^l} \mathbb{E}[\mathbf{W}^l_{j,i,g,h}\phi(\mathbf{A}^l_{i,h}(\mathbf{S}))] = 0 \tag{3.11}$$

$$\begin{aligned} &cov\left[\mathbf{W}^l_{j,i,g,h}\phi(\mathbf{A}^l_{i,h}(\mathbf{S})), \mathbf{W}^l_{j,i',g,h'}\phi(\mathbf{A}^l_{i',h'}(\mathbf{S}^{'}))\right] \\ &= \sigma_b^2 + \sigma_w^2 \sum_{h \in g\text{th patch}} \mathbb{E}\Big[\phi(\mathbf{A}^l_{i,h}(\mathbf{S}))\phi(\mathbf{A}^l_{i,h}(\mathbf{S}^{'}))\Big], \end{aligned} \tag{3.12}$$

where $\mathbb{E}[.]$ represents the mathematical expectation and $cov[.]$ represents the element-wise covariance function. The Gaussian distribution on the weights and feature maps is assumed to be with a zero mean. The convariance includes a term depending on $\phi(\cdot)$. According to [6], a closed form of the covariace can be obtained if $\phi(\cdot)$ is Gaussian and ReLU etc. Please refer to [6] for a solution with ReLU activation. With equation (3.11) and (3.12), a deep convolutional neural network can be considered as a GP. A deep convolutional neural network is well known for spatial feature extraction, while GP is mostly famous for temporal data regression. In this thesis, a ConvNet GP is proposed by equating the base-line CNN as shown in Table 3.2 to a GP. By incorporating a CNN into the kernel of a GP, the powerful feature extraction ability of the CNNs is embedded into GP, and the GP is powerful Bayesian framework that can analyse uncertainty information. The equivalence between CNNs and GPs preserves the advantages of both.

The next section introduces another proposed framework that equips the CNN with the ability of uncertainty quantification.

3.2.3 Performance Evaluation

Data Pre-processes and Preparation: Traffic speed data used is collected on road segments in the city centre of Santander with 15-minute time steps for the year 2016.

The traffic dataset is provided thanks to the SETA EU project. The traffic speed data structured in equation (2.24) are divided as follows to prepare the training, validating and testing set. Assume that each sample comprises S rows, and 10 samples constitute one loop of data partition.

- Training Set: The first 7 samples (1-7)
- Validating Set: The next 2 samples (8-9)
- Testing Set: The next 2 samples (10)

In this way, unique features of traffic data, such as those during Christmas vacation can be captured. CNN, CNN with GP Regression and ConvNet are described in Section 3.2. Each proposed network is performed to accomplish the following five tasks:

- Task 1: 1-step ahead prediction, with 10-step traffic speed history on 20 road segments.
- Task 2: 2-step ahead prediction, with 10-step traffic speed history on 20 road segments.
- Task 3: 1-step ahead prediction, with 14-step traffic speed history on 50 road segments.
- Task 4: 2-step ahead prediction, with 14-step traffic speed history on 50 road segments.
- Task 5: Based on Task 1, different levels of simulated sensor noise are added to the data, and therefore the noisy data become $\mathbf{X}_{noisy} = \mathbf{X} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\mu}_{noise}, \boldsymbol{\sigma}^2_{noise})$.

The models are implemented on Python by using Tensorflow, Keras and GPflow. The training and performance evaluation are run on a PC with 8-core i7-10700K CPU, 48 GB memory, and an RTX-2080 GPU. CNN took 11 ± 0.6 seconds to train and evaluate once, CNN with GP regression took 50 ± 9 seconds to train and evaluate once, and

ConvNet took 1.3 ± 0.2 seconds to train and evaluate once.

Evaluation and Dissuasion: In the application of CNN and CapsNet, both networks employ mean squared error (MSE) as the loss function. Adam optimiser [104] with the exponentially decaying learning rate is utilised to minimise the total MSE. For implementation of CNN as shallow GP, the ConvNet GP is the kernel equivalent to a 6-layer CNN that has three convolutional layers and three max pooling layers. The parameters and layer settings are the same as the typical CNN whose architecture and layer parameters are listed in Table 3.2. The layer parameters for CapsNet are listed in Table 3.3.

Table 3.2: *Layer parameters of CNN*

Layer	Parameter	Activation
Convolution1	(16,3,3)	ReLu
Polling1	(2,2)	-
Convolution2	(8,3,3)	ReLu
Polling2	(2,2)	-
Convolution3	(4,3,3)	Relu
Polling3	(2,2)	-
Flatterning	-	-
Fulling-connected	-	-

Table 3.3: *Layer parameters of CapsNet*

Layer	Parameter	Activation
Convolution1	(32,3,3)	ReLu
Convolution2	(32,3,3)	ReLu
PrimaryCaps	(128,3,3)	Relu
	Capsule size 8	-
TrafficCaps	Capsule size 16	-

The root mean squared error (RMSE) as a measurement of the accuracy of traffic speed prediction performance. The RMSE is defended as:

$$RMSE = mean(\sqrt{\frac{\sum_{i=1}^{I}(y_i - \widehat{y_i})^2}{N}}) \tag{3.13}$$

where y_i is the ground truth value; $\widehat{y_i}$ represents the traffic speed prediction on i-th road segment; and N denotes the number of the number of the traffic speed data in

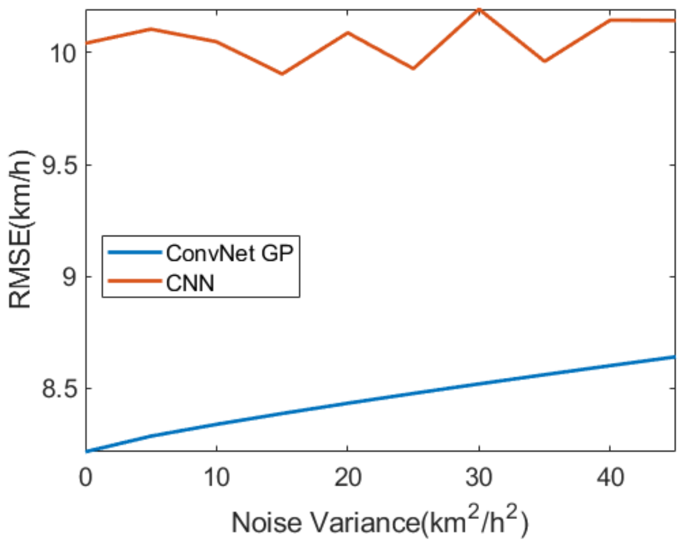
Table 3.4: *Results validation (unit: km/h)*

	CNN RMSE	CapsNet RMSE	ConvNet GP RMSE
Task 1	8.94	8.853	6.69
Task 2	9.39	9.179	6.97
Task 3	9.70	9.257	8.02
Task 4	9.84	9.472	8.205

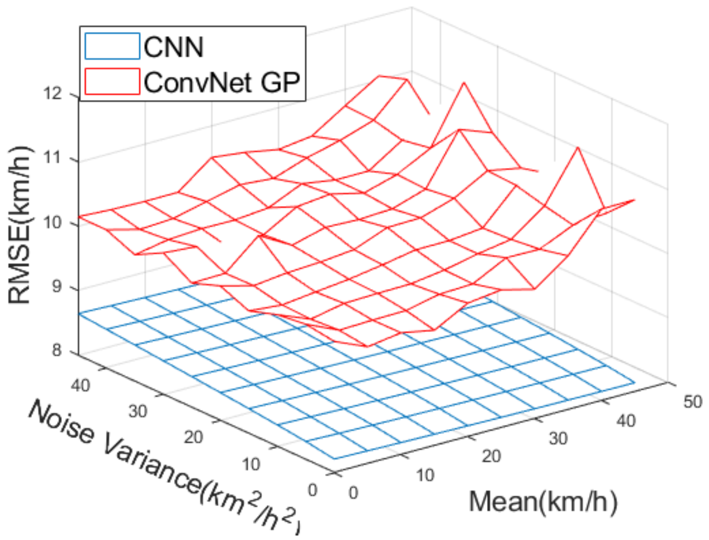
evaluation set. The performances of the networks operating four tasks are listed in Table 3.4.

The RMSE increases as input and output sizes increase from Task 1 to Task 4. CNN proposed by Ma et al. [45] is the base line. CapsNet contributes to the improvements of 1. 0%, 2. 3%, 4. 6% and 3. 8% compared to CNN with respect to Task 1 to Task 4. Among the four approaches introduced in previous sections, ConvNet GP provides the best performance on all four tasks, as it shows better (smaller) RMSE than the other four approaches, especially for Task 1 and 2. ConvNet GP provides the most significant improvement, 25.2% and 25.8% smaller, in RMSE when performing Task 1 and 2 compared to CNN. For Task 3 and 4, ConvNet GP contributes a 17.3% and 16.7% improvement.

Fig. 3.5 shows the overall performance of networks operating with different noise levels (Task 5) where the noise variance ranges from 0 to 45. For ConvNet GP, the RMSE values increase as the noise variance increases. Fig. 3.6 shows that the performances on each sensor with different noise levels are consistent with the overall performance. ConvNet GP provides the smallest RMSE values on all sensors, and the RMSE values increase as the noise variance increases. The performance of the networks maintain



(a)



(b)

Figure 3.5: *a) Validation on data with zero mean noise. b) Validation on data with non-zero mean noise. Blue represents the RMSE value obtained from ConvNet GP; Red represents the RMSE obtained from CNN.*

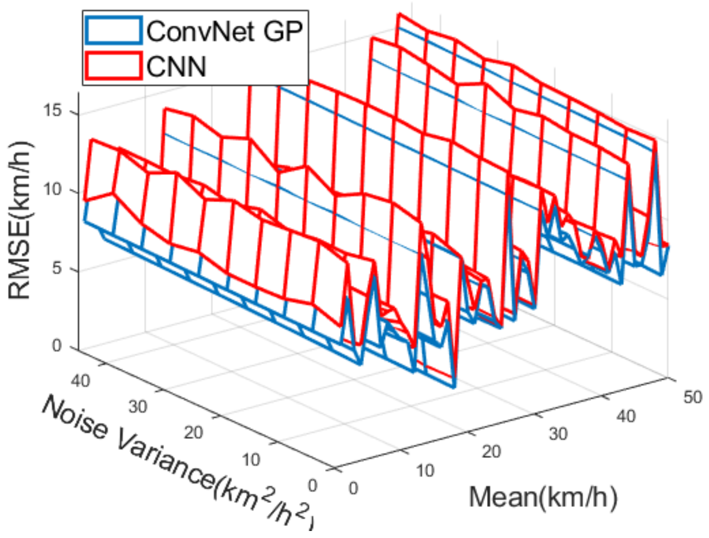


Figure 3.6: *Validation on data with zero mean noise for each sensor. Blue represents the RMSE value obtained from the ConvNet GP; Red represents the RMSE obtained from the CNN.*

the same tendency as the sensor ID varies. For example, the largest RMSE values are located on the sensor 26 for all networks, and the smallest RMSE values are located on the sensor 18 for all networks.

Fig. 3.7 shows the confidence interval of GP and Convnet GP with 3σ in purple. The blue dots represent the ground truth. Therefore, the ConvNet GP provides a very narrow confidence interval. To conclude, the ConvNet GP provides the best accuracy with respect to the lowest RMSE, but provides the narrowest confidence interval.

3.2.4 Conclusion

This section shows that the properties of Gaussian process regression can be beneficial in assessing the impact of sensor data uncertainties in traffic speed prediction. The section presents a GP based frameworks, and ConvNet GP. The ConvNet GP reformulates a CNN as a Gaussian process. The ConvNet GP is compared to a baseline generic CNN. The real traffic speed data collected in Santander city, Spain, is used to validate

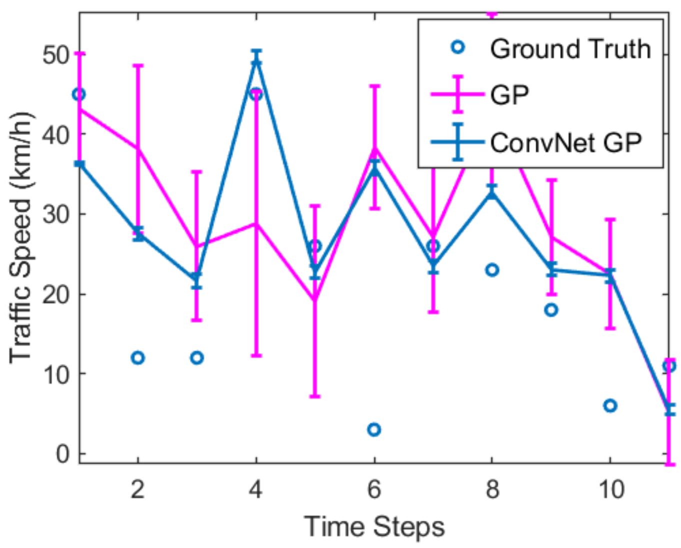


Figure 3.7: *Prediction uncertainty obtained by ConvNet GP. The 1σ confidence interval is applied on GP and 3σ confidence interval is applied for ConvNet GP.*

and evaluate the performance of the proposed GP frameworks. The results imply that the ConvNet GP has better capabilities on learning in the presence of uncertainties in the test data and gives 18.23% improvement in the speed RMSE with respect to the generic CNN. With the help of variances provided from the GP, the calculated confidence intervals characterise the credibility of the speed predictions. This section evaluates the performance with different levels of noise in data, which shows that the ConvNet GP achieves the best performance over all levels of noise and the RMSE increases as the noise level increases for all the methods.

3.3 A Convolutional Neural Network - Gaussian Process with Deep and Multitype Kernels for Time Series Prediction

Time series prediction is a fundamental problem in various fields, ranging from stock price and economic monitoring to traffic flow prediction and global environment forecasting. Time-series prediction models forecast future values or patterns based on historical observations, enabling researchers and practitioners to make better decisions. Time-series prediction is essentially a regression problem that involves understanding and modelling the relationship between independent variables and the dependent variable. Numerous algorithms have been proposed for time series regression, including ARIMA [28], LSTM [3], and CNN [45].

Despite the success of existing regression models in mapping inputs to feature spaces [105, 106, 107, 108, 109], they cannot provide uncertainty information. Uncertainty quantification (UQ) plays a crucial role in decision making and prediction, enabling the rigorous analysis of inherent uncertainties within models and data. UQ involves the use of statistical methods to quantify the uncertainty in the outputs of complex mathematical models. Without UQ, the decisions and predictions made are usually not trustworthy [60, 110]. Bayesian techniques are successfully demonstrated frameworks that learn uncertainty [110], including Monte Carlo (MC) dropout [111], Markov chain Monte Carlo (MCMC) [112], and variational inference (VI) [113]. These techniques provide a probabilistic framework for modelling uncertainty, enabling the incorporation of prior knowledge and updating beliefs as new data is observed.

A Gaussian Process (GP), being among the most powerful tools in the Bayesian inference regime, has the potential to equip CNNs with the ability for uncertainty analysis. A GP is a collection of random variables, any finite number of which has a joint Gaussian distribution [56]. GPs have received increased interest due to their flexibility, robustness, and wide range of applicability. One of the key advantages of GPs is that they are non-parametric, meaning that they can flexibly adapt to complex data without the need to manually specify the latent functions. The most important advantage of a GP is its ability to provide a measure of uncertainty, which is particularly useful in applications where it is crucial to estimate not only the outcomes but also the associated uncertainty.

To fill the gap in existing regression models with CNN embedding, GP regression is introduced. In this paper, CNN-GP and a spatio-temporal kernel are proposed for time-series prediction. CNN-GP incorporates a CNN feature embedding and a GP regression. The spatio-temporal kernel of the GP provides consideration for both time and space, which can bring improvements in performance.

The contributions of this section are the followings:

- A CNN-GP framework for time series prediction is proposed. The CNN provides feature maps to the GP regression which makes the prediction and quantify the impact of uncertainties;
- A multitype spatio-temporal GP is proposed to better encode the prior knowledge observed from the data.
- A detailed performance validation and evaluation of these proposed approaches is conducted on two case studies: (1) seasonal carbon dioxide (CO_2) prediction, (2) traffic prediction in terms of volume and speed data.
- The proposed approach with spatio-temporal kernels can provide both short-term and long-term predictions.

The next section proposes the CNN-GP framework and spatio-temporal kernel.

3.3.1 Related work

This section briefly reviews commonly used models for time-series prediction. Primarily, regression models are divided into two categories: model-based and data-driven methods [114, 115]. The main difference between the two methods is, as their names suggest, whether models or data are used to predict future states. Model-based methods require the development of suitable models that describe the dynamics of systems. The ARIMA algorithm [28] was proposed in 1979 and achieved success in short-term highway traffic prediction, using a physical traffic model. Since then, model-based methods have attracted the interest of researchers. With the assumption that models can accurately describe the dynamics of the traffic system, the prediction results based on physical models are reliable. While model-based methods focus on building physical models [20, 28], data-driven methods mostly require only historical data. Statistical and machine learning methods are the two main categories that have been developed to investigate the inherent relationships in data. Statistical data-driven methods, such as the support vector machine (SVM), have achieved success with time series data.

ANNs are also widely applied to time series predictions due to their ability to work with multidimensional data and their generalisability [33]. Park et al. [34] proposed a real-time vehicle speed prediction algorithm based on an ANN. Zheng et al. [35] combined Bayes' theorem with an ANN to predict short-term freeway traffic flow. However, the data-driven mechanism of a simple ANN cannot explain the spatial relationships of the road segments. Moreover, compared to deep learning methods, ANNs provide lower prediction accuracy due to their shallow architectures. To capture comprehensive traffic flow features both spatially and temporally, a deep-learning-based model, the stacked restricted RBM, is proposed [29]. The RBM model is employed to extract these traffic flow features, and subsequently, a logistic regression approach is utilised for the forecasting process. Most recently, an adaptive spatio-temporal (AST) InceptionNet was proposed [116]. By leveraging the inception module, the model effectively combines local spatio-temporal features with various global features. Additionally, the development of a fully adaptive graph convolution method, encompassing topologically adaptive graph convolution and an adaptive adjacency matrix, enables autonomous and dynamic learning of spatial heterogeneity. This allows the AST-InceptionNet to effectively handle situations where adjacency relations are unknown. Moreover, a long-short temporal information fusion module is proposed [117], which integrates a transformer network and a spatio-temporal graph convolutional network.

However, real-world phenomena often encompass numerous confounding factors and involve a large number of inputs. High-dimensional data can be redundant or highly correlative [118, 119]. This necessitates the utilisation of mapping inputs into feature spaces with reduced dimensionality by transforming the high-dimensional input space into a lower-dimensional feature space. Through the representation of data in a more abstract form, mapping inputs into feature spaces effectively reduces computational complexity while enhancing the efficiency of regression models. Simultaneously, mapping inputs into feature spaces facilitates the extraction of informative features from the input data. By capturing the underlying patterns and relationships inherent within the data, mapping the input into feature spaces enables the regression model to focus on the most salient and informative features. Consequently, the regression model can better capture the underlying dependencies and improve the prediction performance. This approach is widely applied in neural language processing tasks [105, 106] and logistic regression [107]. From another perspective, the relationships between inputs and outputs frequently exhibit non-linear characteristics. In this regard, mapping inputs into feature spaces, particularly those based on deep learning models, are able to capture non-linear dependencies and, consequently, improve the accuracy of predictions. Various mapping methods have been proposed, including common methods such as linear discriminant analysis [120, 121], principal component analysis [91, 122], and independent component analysis [123], among others [124, 125]. However, these methods are considered single layer learning methods [126]. Some deep models have been proposed, for example, the stacked autoencoder (SAE) [127] and DBN [128, 41]. These deep models extract robust features and outperform other methods. However, SAE and DBN are not able to extract spatial information effectively. The CNN employs local connections to extract spatial information efficiently and shared weights to substantially diminish the number of parameters. A hybrid deep neural network model (HDNN) [108] has been proposed for volatility forecasting. The HDNN consists of feature extraction and regression components. Mapping inputs into feature spaces explores the volatility-related feature space via a CNN, and the fully connected network, considered as the regression component, takes the feature vectors to predict volatility, bringing remarkable improvements in prediction accuracy. Another hybrid framework has been proposed for diabetic retinopathy classification [109], where a deep CNN-based architecture is used as the feature extractor and the extracted feature vectors are classified by a support vector machine, which reduces misclassifications. The CNN feature extraction has shown powerful feature extraction ability and can potentially be combined with different regression and classification models.

In addition to neural networks, kernel methods are also popular in time-series prediction. Kernel methods are considered as non-parametric models, In the context of machine learning, a non-parametric model is one that does not make strong assumptions about the form or structure of the function that is being estimated. Instead, kernel methods use kernel functions to map the data into a higher-dimensional space where the learning problem can be solved more easily. One of the most well-known kernel methods is SVM. In [129], an SVM is proposed for financial time series forecasting. GPs are a form of the kernel method and were popularised by Rasmussen and Williams [56]. Xie et al. [130] proposed a GP for short-term traffic volume prediction. Inspired by mapping inputs into feature spaces, deep kernel learning (DKL) is proposed to combine the strength of deep learning and kernel methods. DKL maps the inputs to an intermediate space through a deep neural network, such as a deep CNN, and these intermediate values are used as inputs to standard kernel methods [131, 132].

However, neural networks lack uncertainty analyses. MC dropout [111] is a regulari-

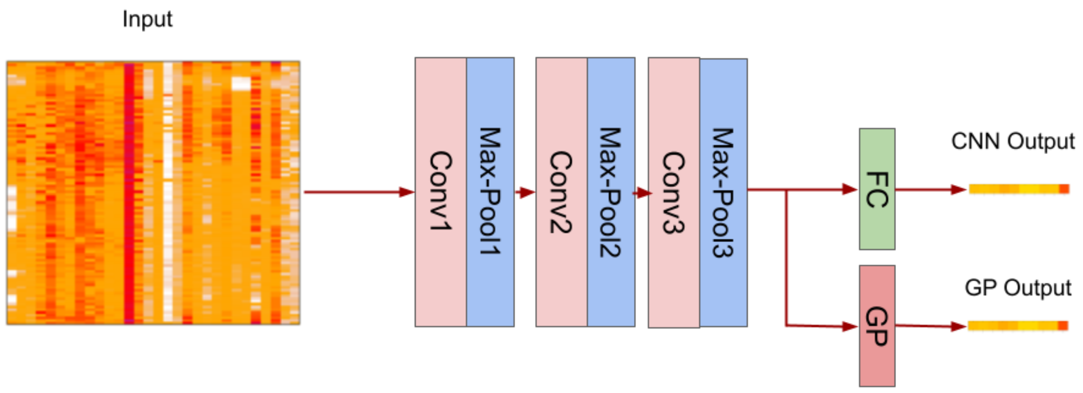


Figure 3.8: *The GP-CNN framework. Blue block represents the combination of convolutional and pooling layers; red block represents fully connected layer; and pink block represents a Gaussian process regression.*

sation technique introduced in deep learning models for UQ. MC dropout enables the model to sample multiple predictions for a given input by randomly dropping out neurons during both the training and inference stages, allowing the deep learning model to estimate uncertainties. As proposed in [133], MC dropout with input-dependent data noise is applied to quantify the uncertainty for LSTM models. MCMC [112] is a powerful statistical technique in Bayesian inference. It provides an approach to explore high-dimensional space and estimate posterior distributions, which can generate representative samples that approximate the uncertainty quantification and desired distribution. Another Bayesian statistical technique for quantifying uncertainty is VI. VI offers computationally efficient frameworks to approximate intractable posterior distributions [113]. In particular, VI is often used to estimate the posterior distribution of the neural network parameters. The estimated parameter distributions enable the model to quantify uncertainty. Furthermore, VI allows for the calculation of additional quantities, such as the evidence lower bound. The confidence interval (CI) is one of the most common UQ methods. CI is a statistical tool that is used to estimate the distribution based on a sample from that population, providing a measure of uncertainty.

3.3.2 CNN with Spatio-temporal GP Regression

A spatio-temporal GP (STGP) is a GP model with spatial-temporal kernel function that models the evolution of the system in both space and time [134]. In this section, the spatial inputs for the GP framework are the past observations, which is represented as x, and the temporal inputs are the time, t. An STGP can be used to model a functional transformation from the inputs to the output Y [135]:

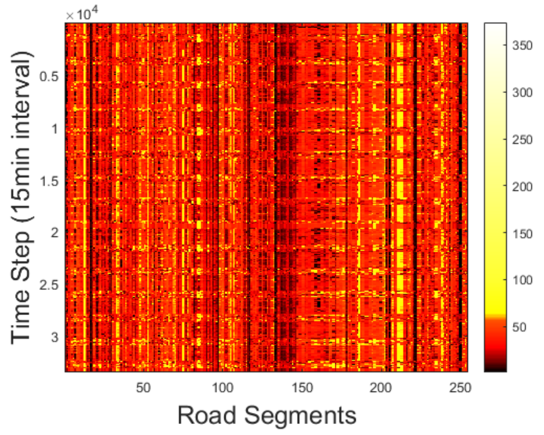
$$\begin{aligned} Y &= f(\mathbf{x}, t), \\ f(\mathbf{x}, t) &\sim \mathcal{GP}(m(\mathbf{x}, \mathbf{x}), k(\mathbf{x}, \mathbf{x}'; t, t')), \end{aligned} \tag{3.14}$$

where $m(\cdot)$ and $k(\mathbf{x}, \mathbf{x}'; t, t')$ are, respectively, mean function and spatial-temporal kernel (covariance) function. The kernel function can be factorised as:

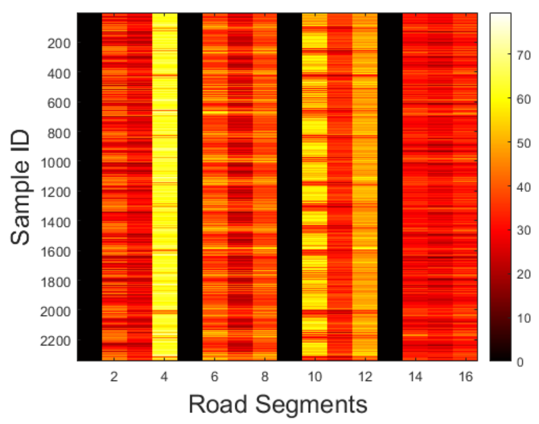
$$k(\mathbf{x}, \mathbf{x}'; t, t') = k_s(\mathbf{x}, \mathbf{x}')k_t(t, t'), \tag{3.15}$$

where $k_s(\cdot, \cdot)$ and $k_t(\cdot, \cdot)$, respectively, represent the spatial and temporal kernel functions. With the mean and covariance function, the STGP regression can be formulated in the same way as the GP described in the previous section. The complexity of the GP for N observations is $\mathcal{O}(N^3)$, and the complexity of the STGP for N observations and T time steps is $\mathcal{O}(N^3T^3)$.

Proposed in [136], a Gaussian process can be used in conjunction with a CNN to analyse uncertainty without decreasing accuracy. The primary concept behind CNN with GP regression is to abstract features using a combination of convolutional and pooling layers, and then input the abstracted features into a GP regression model to generate the predictions. As shown in Fig. 3.8, a pre-trained CNN is used to abstract features and the last-layer features of the CNN are fed into the GP regression to generate prediction and uncertainty quantification. Therefore, the input of GP, $\mathbf{z}$, is the last-layer features of the CNN. For example, Fig. 3.9 a) shows a input of traffic speed



(a)



(b)

Figure 3.9: *a) The visualization of the traffic speed data. b) The visualization of the last-layer features extracted by the typical CNN.*

prediction and Fig. 3.9 b) presents the last-layer features of the CNN. From equation (2.27), let $\mathbf{z}$ denote the last-layer features fed into the GP regression. Therefore, we have $f(\mathbf{z}) \sim \mathcal{GP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}'))$ and $f(\mathbf{z}, t) \sim \mathcal{GP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}'; t, t'))$. Following equations (2.61)-(2.66), the mean and covariance of the prediction distribution can be calculated.

3.3.3 Experiments And Analysis

3.3.3.1 Index of Performance

The root mean squared error (RMSE) is the main performance evaluation metric used to evaluate traffic speed prediction approaches. The RMSE is defined as follows:

$$RMSE(k) = \sqrt{\frac{1}{N_{mc}} \sum_{m=1}^{N_{mc}} [\hat{y}(k) - y(k)]^2}, \tag{3.16}$$

where $y(k)$ is the ground truth speed value and $\hat{y}(k)$ represents the traffic speed prediction at the kth time step. Here N_{mc} denotes the number of Monte Carlo runs.

Table 3.5: *Layer parameters of CNN for CO_2 prediction*

Layer	Parameter	Activation
Convolution1	(32,3)	ReLu
Polling1	(2,2)	-
Convolution2	(16,3)	ReLu
Polling2	(2,2)	-
Convolution3	(8,3)	Relu
Polling3	(2,2)	-
Flattern	-	-
Fulling-connected	-	-

Table 3.6: *Layer parameters of CNN for traffic volume prediction*

Layer	Parameter	Activation
Convolution1	(64,3)	ReLu
Polling1	(2,2)	-
Convolution2	(32,3)	ReLu
Polling2	(2,2)	-
Convolution3	(16,3)	Relu
Polling3	(2,2)	-
Flattern	-	-
Fulling-connected	-	-

3.3.3.2 Seasonal CO_2 Prediction

The seasonal CO_2 measurements are collected at the Mauna Loa Observatory in Hawaii [137]. The CO_2 data collection was started by Charles David Keeling in March 1958 [138]. The dataset contains monthly average CO_2 values from 1958 to 2001, and each monthly average is the mean of daily averages. There are 521 time steps, and the data contains times and atmospheric CO_2 values. Therefore, the dataset is a vector for values $\mathbf{C} = [C_1, \cdots, C_{510}]$ and a vector for time $\mathbf{T} = [T_1, \cdots, T_{510}]$. We use the first 357-month history (70%) as training data and the remaining 153-month history (30%) to evaluate the proposed approach. In this case study, CNN and CNN-GP described in previous sections perform the following tasks:

- Short-term task: One-month prediction, with 10-month history. For each prediction, actual data is used to form the history.

- Long-term task: 24-month prediction, with 357-month history. For each prediction after the first within the 24-month period, we use the predictions from previous steps to form part of the history.

For short-term prediction, a 10-month history is used jointly with the time to train the CNN and CNN-GP model. Therefore, the input of the CNN and the CNN module in the CNN-GP framework can be represented as $\mathbf{C} = [c_1, \cdots, c_{10}]$, and the input fed into the GP regression can be represented as $\mathbf{z} = [z_1, \cdots, z_n, t]$. To evaluate the models, 20 experiments are conducted to compute the performance of the baseline CNN and the CNN-GP framework with three different combinations of kernels. The CNN architecture consists of three pairs of convolutional and pooling layers followed by a flattening layer and a fully connected layer. Table 3.5 defines the architecture of the CNN and layer parameters for CO_2 concentration prediction. The three convolutional layers have 64, 32, 16 channels, respectively, of size 3 with a stride of 1. Each convolutional layer involves a rectified linear unit (ReLU) activation function. Pooling layers have filters of size 2 applied with a stride of 1. The architecture of the CNN-GP framework is represented in Fig. 3.8. The different options for the GP module kernel function are listed in (3.17) to (3.19):

$$k_1(\mathbf{z}, \mathbf{z}') = k_{SE}(\mathbf{z}, \mathbf{z}') + k_{Linear}(t, t'), \tag{3.17}$$

$$k_2(\mathbf{z}, \mathbf{z}'; t, t') = k_{SE}(\mathbf{z}, \mathbf{z}') + k_{PerSE}(t, t') + k_{Linear}(t, t'), \tag{3.18}$$

$$k_3(\mathbf{z}, \mathbf{z}'; t, t') = k_{SE}(\mathbf{z}, \mathbf{z}') \times k_{PerSE}(t, t') + k_{Linear}(t, t'), \tag{3.19}$$

where k_{SE} represents the square exponential (SE) kernel, and k_{Linear} denotes the linear kernel. The periodic kernel k_{PerSE} can be achieved by mapping the original inputs through the transformation $\mathbf{u} = [\sin t, \cos t]$. In the context of CO_2 concentration prediction, the periodic kernel takes the SE kernel as the base kernel, and the formulation can be expressed as:

$$k_{PerSE}(t, t') = \sigma_f^2 \exp\left(-\frac{\sin^2 \frac{\pi|t-t'|}{p}}{2l^2}\right), \tag{3.20}$$

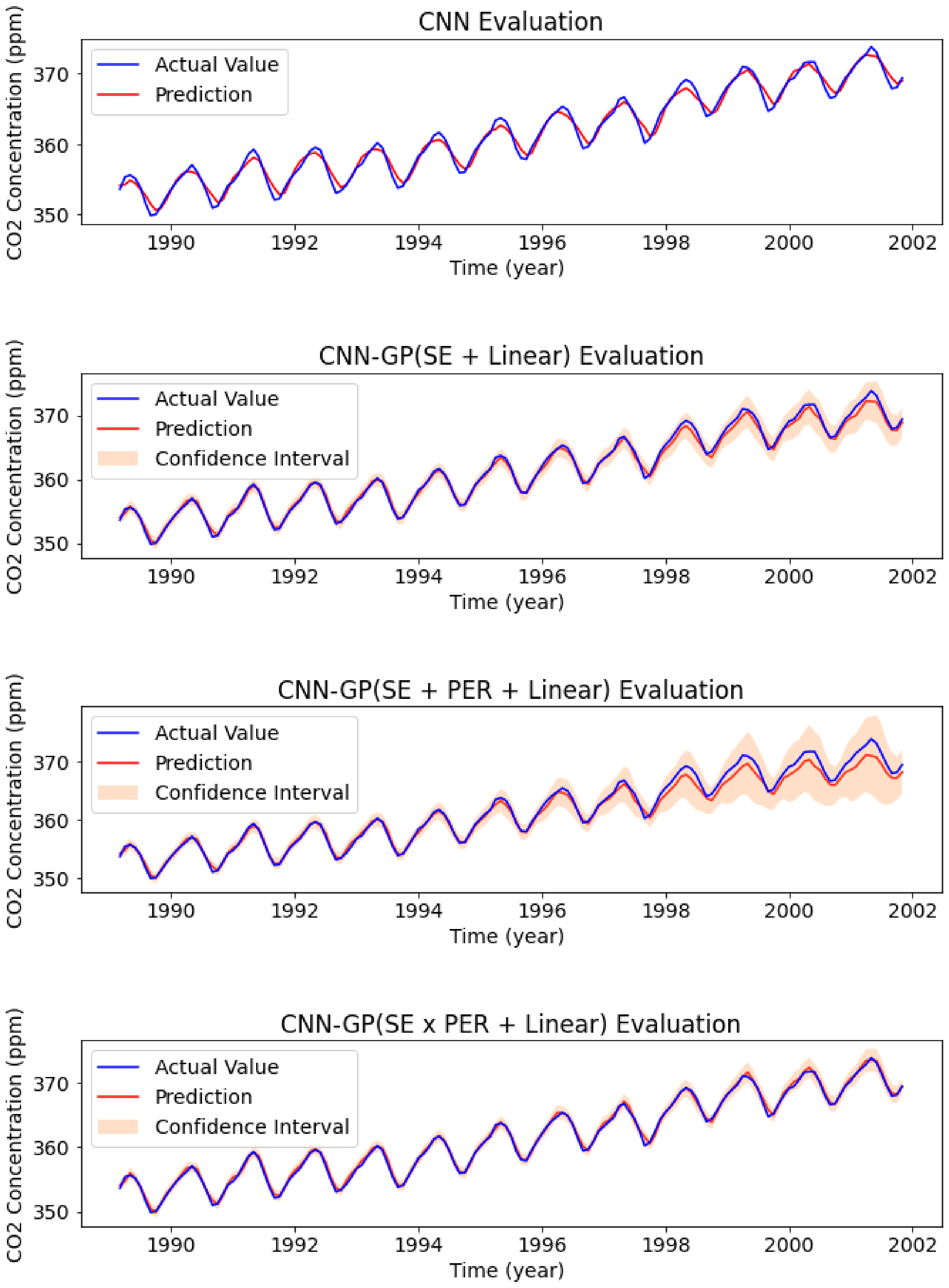


Figure 3.10: *Visualization of short-term (1 month) CO_2 concentration prediction for the CNN and CNN-GP with different spatio-temporal kernels.*

where l and σ_f denotes length-scale and variance respectively, and p represents the period parameter. The linear kernel is defined as:

$$k_{Linear}(t, t') = vtt', \tag{3.21}$$

where v is a coefficient.

The results are presented in Fig. 3.10, where the red line represents the average prediction over 20 experiments, and the blue line represents the actual CO_2 concentration. The CNN model, with the parameter settings outlined in Table 3.5, achieves an RMSE of 1.719 parts per million (ppm). The CNN-GP framework, with the kernel shown in equation (3.17), achieves an RMSE of 0.866 ppm; the CNN-GP framework, with the kernel shown in equation (3.18), achieves an RMSE of 0.849 ppm, excluding one of the experiments that did not converge; and the CNN-GP framework, with the kernel shown in equation (3.19), achieves an RMSE of 0.428 ppm. As shown in Fig. 3.10, the CNN-GP with the spatio-temporal kernel shown in equation (3.19) provides the most stable and accurate predictions. The experimental results demonstrate that the CNN-GP framework with the spatio-temporal kernel exhibits the capability to generate accurate predictions, even when the employed CNN model does not perform well. The peach shadows in Fig. 3.10 represent the 3σ confidence interval, which means that the CNN-GP framework has 95.4% confidence that the actual observations are located within the interval. This demonstrates an advantage of the CNN-GP method: confidence intervals are available, whereas pure CNN does not produce CIs. For long-term prediction, a 357-month history is used to train the model, and predictions are made for 24 months using the predictions from the previous steps. The input of the CNN and the CNN-GP framework can be denoted as $\mathbf{C} = [\hat{c}_1, \cdots, \hat{c}_{10}]$, where $\hat{c}_n$ represents the predictions made by the model in the previous steps. The spatio-temporal kernel used for the CNN-GP framework is shown in equation (3.19). Fig. 3.11 shows the results for long-term prediction, where the red line represents the long-term predictions and the blue line represents the actual CO_2 concentration. CNN achieves an RMSE of 1.524 ppm, and the CNN-GP framework achieves 0.740 ppm. Fig. 3.11(b) presents

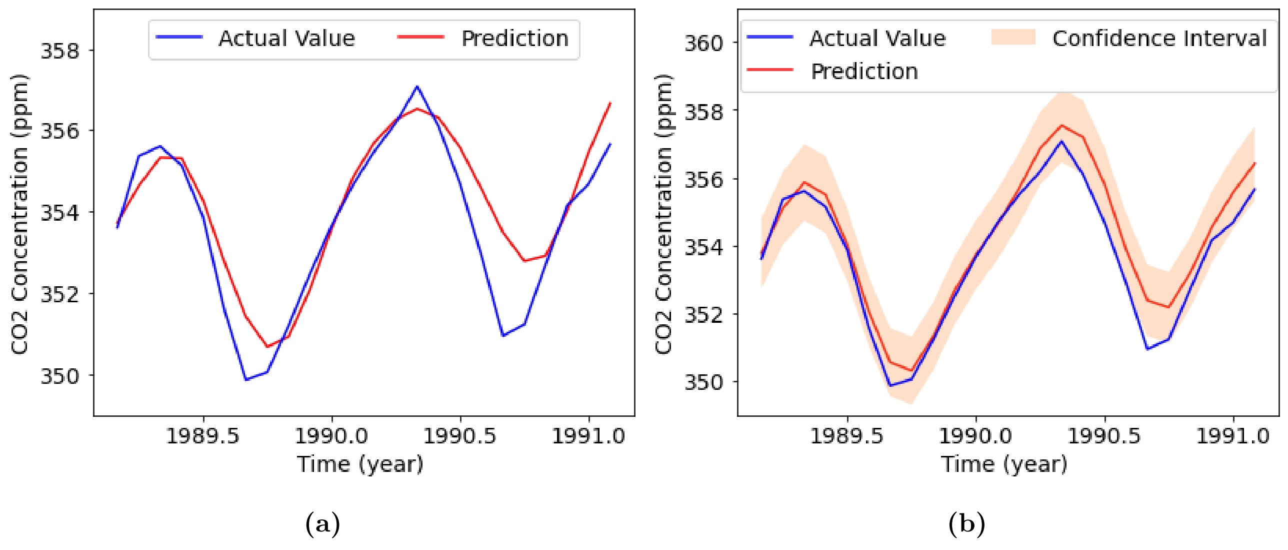


Figure 3.11: *a) Visualization of long-term (24 months) CO_2 prediction by using CNN. b) Visualization of the long-term CO_2 prediction by using CNN-GP with spatio-temporal kernel as shown in equation (3.19) and uncertainty information evaluated by confidence interval (3σ).*

the uncertainty quantification of the CNN-GP framework with a confidence interval. As shown in the figure, the peach shadows represent the 3σ confidence interval, meaning that the CNN-GP framework has 99.7% confidence that the actual observations are located within the interval. As with short-term predictions, the CNN-GP outperformed the CNN. Due to different time periods used in the assessment, it is not possible to compare the short and long-term predictions. However, long-term predictions are expected to be worse than short-term ones.

3.3.3.3 Traffic Prediction

The traffic dataset is provided thanks to the SETA project [139]. The traffic data we use is collected on road segments in the city centre of Santander with 15-minute time steps for the year 2016. There are $N = 33504$ time steps, excluding days when the sensors did not work, and $M = 256$ road segments. Therefore, the traffic data are a matrix of size $N \times M$. Each missing measurement is estimated using an average of the

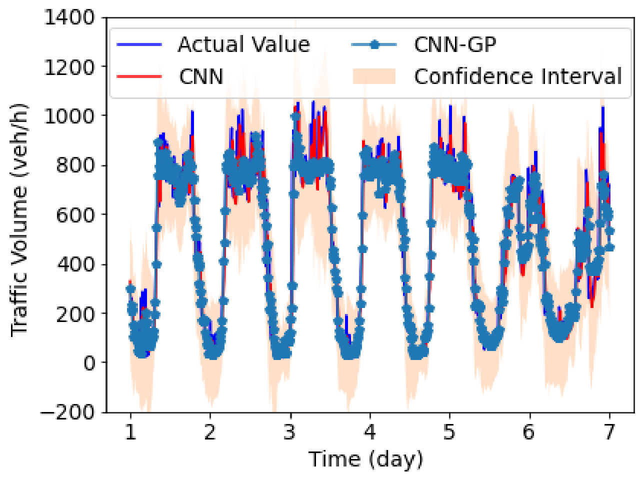


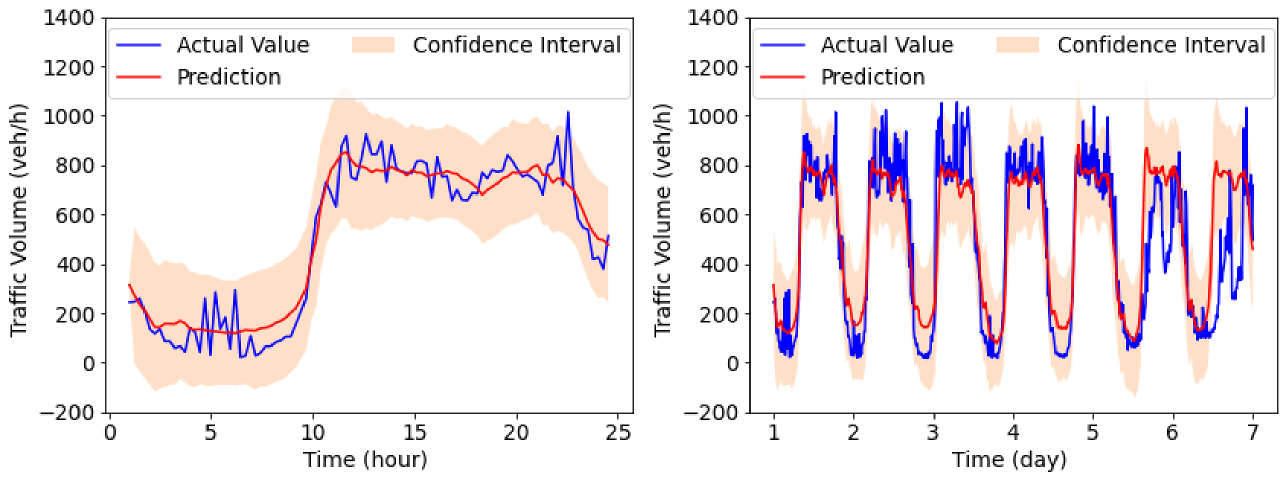
Figure 3.12: *Visualization of short-term (15 minutes) traffic volume prediction for the CNN and CNN-GP with spatio-temporal kernel. Note that the predicted traffic volume values are positive but the negative values are due to the* 3σ *confidence intervals.*

traffic at the same time on the other days. Therefore, the traffic volume data, **V**, and speed data, **S**, can be represented in an image-like matrix form:

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N1} & v_{N2} & \cdots & v_{NM} \end{bmatrix}, \tag{3.22}$$

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{NM} \end{bmatrix}. \tag{3.23}$$

The pixel value of v_{ij} and s_{ij} represents the traffic volume and speed on the road segment j at time step i. Therefore, the **V** and **S** matrix each form a channel of the image. We use 2-week history (1344 time steps) as a training set and use 1-week traffic to evaluate the proposed approach.



(a) **(b)**

Figure 3.13: *a) Visualisation of long-term (1 day) traffic volume prediction. b) Visualisation of long-term (1 week) traffic volume prediction. Note that the predicted traffic volume values are positive but the negative values are due to the* 3σ *confidence intervals.*

In this case study, CNN and CNN-GP described in previous sections, respectively, perform the following task:

- 1-step (short-term) traffic volume prediction on one specific road segment, with 10-step history on the specific road segments. For each new prediction, we provide accurate historical data.
- 1-week (long-term) traffic volume prediction on one specific road segment, with 2-week history on the specific road segments. For each new prediction, we use the prediction from the previous steps for the history.

3.3.3.4 Traffic Volume Prediction

For short-term traffic volume prediction, a 2-week history of traffic volume is used along with the time to train the CNN and CNN-GP models. The input of CNN and CNN module in the CNN-GP framework can be represented as a matrix $\mathbf{V_i} = [v_{1N}, \cdots, v_{10N}]$, and the input fed into the GP regression can be represented as $\mathbf{z} = [z_1, \cdots, z_n, t]$. CNN and CNN-GP frameworks are trained with a two-week history and tested on the next-week future traffic speed.

The CNN architecture consists of three pairs of convolutional layers and pooling layers, followed by a flattening layer and a fully connected layer. Table 3.6 defines the CNN architecture and the layer parameters for traffic volume prediction. The three convolutional layers have 32, 16, and 8 channels, respectively, with a size of 1×3 and a stride of 1. Each convolutional layer involves a rectified linear unit (ReLU) activation function. The pooling layers have filters of size 1×2 applied with a stride of 2.

The architecture of the CNN-GP framework is represented in Fig. 3.8. The kernel for the GP module is as follows:

$$\begin{aligned} &k(\mathbf{z}, \mathbf{z}'; t, t') = \\ &(k_{M12}(\mathbf{z}, \mathbf{z}') + k_{SE}(\mathbf{z}, \mathbf{z}')) \times k_{PerM32}(t, t') + k_{Bias}(t, t'). \end{aligned} \tag{3.24}$$

where k_{M12} is the Matérn 1/2 kernel, which is equivalent to the exponential kernel, shown in equation (3.25),

$$k_{M12}(\mathbf{z}, \mathbf{z}') = \sigma_f^2 \exp\left(-\frac{|\mathbf{z} - \mathbf{z}'|}{l}\right), \tag{3.25}$$

and k_{PerM32} is the periodic kernel with Matérn 3/2 kernel function [56], formulated as the following,

$$k_{PerM32}(t, t') = \sigma_f^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right), \tag{3.26}$$

where $r = \sin \frac{\pi|t-t'|}{p}$, and the biase kernel is defined as $k_{Bias}(t, t') = c$, where c is a constant. The CNN algorithm, with the parameter settings outlined in Table 3.6, achieves an average RMSE of 73.776 vehicles per hour (veh/h), and the CNN-GP framework, with the kernel shown in equation (3.24), achieves an average RMSE of 75.458 veh/h. Fig. 3.12 presents a visualisation of short-term traffic volume prediction, where the peach shadows represent the 3σ confidence interval, which means that the CNN-GP framework has a 99.7% confidence that the actual observations lie within the interval.

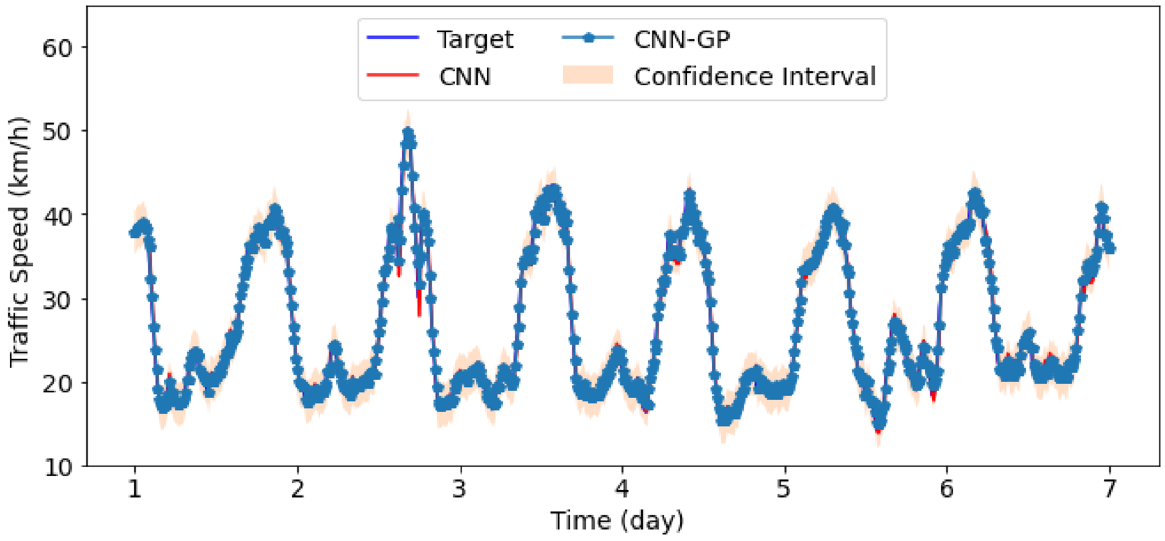
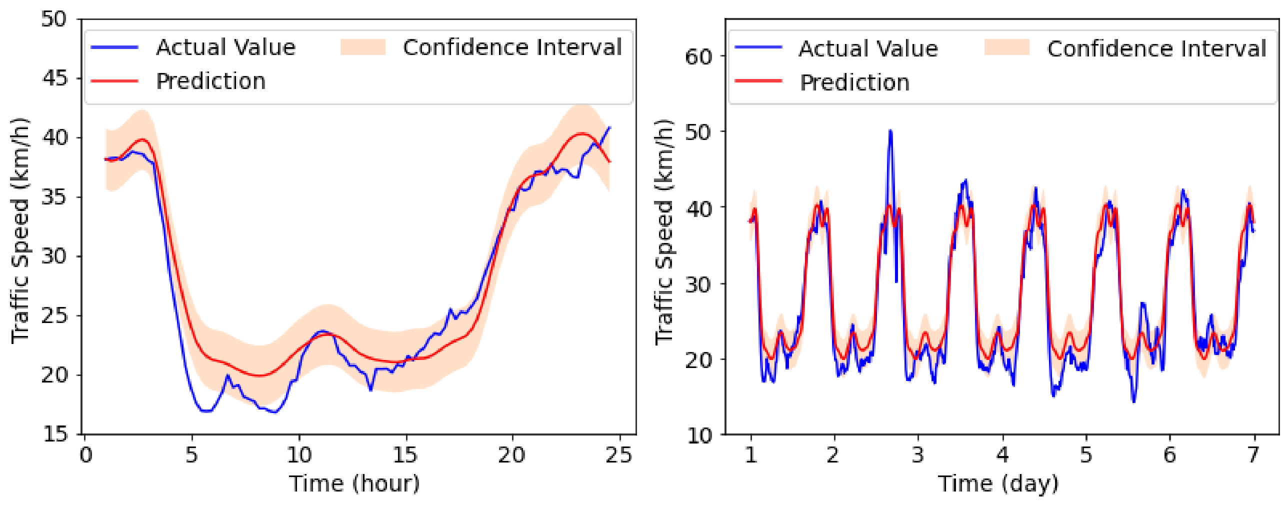


Figure 3.14: *Visualization of short-term (15 minutes) traffic speed prediction for the CNN and CNN-GP with spatio-temporal kernel. Note that the predicted traffic volume values are positive, but the negative values are due to the* 3σ *confidence intervals.*

For long-term prediction, a 2-week traffic volume history is used to train the model, and 1-week predictions are made using the predictions from the previous steps. The input of the CNN and CNN-GP framework can be denoted as $\mathbf{V} = [\hat{v}_{1N}, \cdots, \hat{v}_{10N}]$, where $\hat{v}_{MN}$ represents the predictions made by the model in the previous steps. Fig. 3.13(a) shows the results for 1-day future predictions, and Fig. 3.13(b) presents the results for 1-week ahead predictions. The CNN-GP framework achieves an average RMSE of 130.746 veh/h for 1-day ahead prediction and an average RMSE of 179.872 veh/h for 1-week ahead prediction. The peach shadows shown in Fig. 3.13 represent the 3σ confidence interval, meaning that the CNN-GP framework has a 99.7% confidence that the actual observations lie within the interval.

3.3.3.5 Traffic Speed Prediction

For traffic speed prediction, a 2-week history of traffic speed is used in conjunction with the corresponding time information to train the CNN and CNN-GP models. Therefore,



(a) (b)

Figure 3.15: *a) Visualization of long-term (1 day) traffic speed prediction. b) Visualization of long-term (1 week) traffic speed prediction.*

the input for CNN and CNN module in the CNN-GP framework can be represented as a matrix $\mathbf{S_i} = [s_{1N}, \cdots, s_{10N}]$, where s_{MN} represents the traffic speed at a particular time step. The input fed into the GP regression can be denoted as $\mathbf{z} = [z_1, \cdots, z_n, t]$. The CNN and CNN-GP frameworks are trained using the two-week history and tested on the future week's traffic speed. The CNN architecture is the same as that used for traffic volume prediction, as described in the previous section. The layer parameters, including the number of channels and filter sizes, remain unchanged. The architecture of the CNN-GP framework is illustrated in Fig. 3.8. The kernel for the GP module in the CNN-GP framework is as follows:

$$k(\mathbf{z}, \mathbf{z}'; t, t') = k_{M12}(\mathbf{z}, \mathbf{z}') \times k_{Per_{SE}}(t, t') + k_{Bias}(t, t'), \tag{3.27}$$

where k_{Bias} is the bias kernel. For short-term traffic speed prediction, CNN achieves an average RMSE of 1.144 km/h, while the CNN-GP framework achieves an average RMSE of 0.709 km/h. Fig. 3.14 presents a visualisation of short-term traffic speed prediction, where the peach shadows represent the 3σ confidence interval. This indicates that the CNN-GP framework has a 99.7% confidence level that the actual observations

Table 3.7: *RMSE results summary*

Task	CNN	CNN-GP
CO_2 Short-term	1.719 ppm	0.428 ppm
CO_2 Long-term (24 months)	1.524 ppm	0.740 ppm
Traffic Volume Short-term	73.776 veh/h	75.458 veh/h
Traffic Volume Long-term (1 day)	-	130.746 veh/h
Traffic Volume Long-term (1 week)	-	179.872 veh/h
Traffic Speed Short-term	1.144 km/h	0.709 km/h
Traffic Speed Long-term (1 day)	-	3.796 km/h
Traffic Speed Long-term (1 week)	-	4.190 km/h

fall within the interval.

For long-term prediction, a 2-week history of traffic speed is used to train the model, and predictions are made for the next week using the predictions from the previous steps. The input to CNN and CNN-GP framework can be denoted as $\mathbf{S} = [\hat{s}_{1N}, \cdots, \hat{s}_{10N}]$, where $\hat{s}_{MN}$ represents the predictions made by the model in the previous steps. Fig. 3.15(a) shows the results for 1-day future predictions, and Fig. 3.15(b) presents the results for 1-week ahead predictions. The CNN-GP framework achieves an average RMSE of 3.796 veh/h for 1-day ahead prediction and an average RMSE of 4.190 veh/h for 1-week ahead prediction. The peach shadows shown in Fig. 3.15 represent the 3σ confidence interval, indicating that the CNN-GP framework has a 99.7% confidence level that the actual observations fall within the interval. The results demonstrate that the GP module with a spatio-temporal kernel in the CNN-GP framework has the ability to generate accurate predictions, even when the CNN feature extraction module does not perform well.

3.3.4 Conclusion

This paper introduces a novel deep-kernel CNN-GP framework for time series prediction. The CNN-GP framework combines a CNN module for feature extraction with a GP regression module for prediction and uncertainty quantification. Additionally, a spatio-temporal kernel is proposed for the GP module, allowing it to consider both time and space. Two case studies are conducted to validate and evaluate the performance of the proposed CNN-GP framework. A summary of the results is given in Table 3.7. The experimental results demonstrate that the CNN-GP framework with the spatio-temporal kernel achieves more accurate predictions compared to the baseline CNN model in both CO_2 concentration and traffic prediction tasks. This improvement highlights the effectiveness of incorporating the GP module with the spatio-temporal kernel. It enables the framework to generate accurate predictions, even in scenarios where the CNN feature extraction module may not perform well. Future work will focus on distributed methods for traffic prediction and other methods of uncertainty quantification, e.g. as in [140].

3.4 Uncertainty Quantification with Uniform Error Bound

This section proposes a UEB for CNN-GP to quantify the uncertainty within the traffic speed data. The UEB was proposed based on the GP frameworks, and hence, the prediction variance places a huge impact on the results. The following sections introduce an effectively modified UEB for CNN-GP. As the GP regression takes the last-layer features from the CNN in the CNN-GP framework, the CNN can be considered as the latent function, and the GP regression focuses on modelling the latent function itself. Therefore, the noise term in the GP is removed to generate the UEB for CNN-GP. Performance validation and evaluation for efficient UEB are conducted on simulated and real traffic speed data.

3.4.1 Uniform Error Bound

The UEB is proposed, as data-driven models generate prediction errors due to limited or noisy data. Based on the uncertainty measurement provided by the GP, the UEB is derived with less restrictive assumptions compared with existing error bound methods. First, let $\mathbf{x}$ and $\mathbf{y}$ be the inputs and the targets, respectively, and then $y = f(x) + \epsilon$, where $f(x)$ is the unknown function sampled from a GP with zero mean and ϵ is a zero-mean Gaussian noise with variance σ_n^2 [141]. Recall that a GP is a collection of all possible functions and, therefore, a continuous distribution in the function space can be learnt with discretionary precision by selecting an appropriate kernel function. Besides, the prior over the function space is defined by the GP, and the shape of the prior is typically defined by the kernel function. Therefore, the Lipschitz continuity is required for refining the uniform error bounds. The Lipschitz constant L_k for the kernel function is defined as,

$$L_k = \max_{\mathbf{x},\mathbf{x}' \in \mathcal{X}} \left\| \begin{bmatrix} \frac{\partial \mathbf{K}(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{K}(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_d} \end{bmatrix}^T \right\|. \tag{3.28}$$

However, if limited knowledge of the prior is provided, it is impossible to derive the Lipschitz constant of L_f directly. By assuming a certain prior distribution with kernel function, it is possible to derive the Lipschitz constant. The partial derivatives of a continuous kernel that defines a GP is shown as following,

$$k^{\partial i}(\mathbf{x},\mathbf{x}') = \frac{\partial^2}{\partial \mathbf{x}_i \partial \mathbf{x}_i'} k(\mathbf{x},\mathbf{x}'). \tag{3.29}$$

The Lipschitz constant of the unknown function $f(x)$ is defined as,

$$L_f = \left\| \begin{bmatrix} \sqrt{2\log\frac{2d}{\delta_l}}\max_{\mathbf{x}\in\mathcal{X}}\sqrt{k^{\partial 1}(\mathbf{x},\mathbf{x})} \\ +12\sqrt{6d}\max\{\max\sqrt{k^{\partial 1}(\mathbf{x},\mathbf{x})},\sqrt{rL_k^{\partial 1}}\} \\ ... \\ \sqrt{2\log\frac{2d}{\delta_l}}\max_{x\in\mathcal{X}}\sqrt{k^{\partial d}(\mathbf{x},\mathbf{x})} \\ +12\sqrt{6d}\max\{\max\sqrt{k^{\partial d}(\mathbf{x},\mathbf{x})},\sqrt{rL_k^{\partial d}}\} \end{bmatrix} \right\|, \tag{3.30}$$

where $L_k^{\partial i}$ is the Lipschitz constants of the partial derivative kernels $k^{\partial i}(\cdot,\cdot)$ with maximal extension represented by $r = max_{x,x'\in\mathcal{X}}\|x - x^{'}\|$. Sampled from the GP, the unknown function $f(x)$ is continuous in space $\mathcal{X}$. It has at least $1-\delta_L$ probability that L_f is a Lipschitz constant of $f(x)$ [141]. The core theorem of the UEB can be concluded: the unknown function with the Lipschitz constant L_f is continuous and the observation can be formulated as $y = f(x) + \epsilon$, where $f(x)$ is samped from a GP with zero mean defined by the continuous kernel function $k(\cdot,\cdot)$ with the Lipschitz constant L_k. Next, the mean and covariance functions are continuous with Lipschitz constant L_{vn}.

$$L_{vn} \le L_k\sqrt{N}\|(K+\sigma^2 I_N)^{-1}y_N\| \tag{3.31}$$

The modulus of the continuity can be defined as,

$$\omega_{\sigma N} \le \sqrt{2\tau L_k(1+N\|(K+\sigma^2 I_N)^{-1}\|\max_{\mathbf{x},\mathbf{x}'\in\mathcal{X}} k(\mathbf{x},\mathbf{x}'))} \tag{3.32}$$

$$\beta(\tau) = 2\log(\frac{M(\tau,\mathcal{X})}{\delta}), \tag{3.33}$$

where

$$M(\tau,\mathcal{X}) \le (1+\frac{r}{\tau})^d \tag{3.34}$$

and

$$\gamma(\tau) = (L_v N + L_f)\tau + \sqrt{\beta(\tau)}\omega_{\sigma N}(\tau). \tag{3.35}$$

Then, it holds that

$$P(|f(x) - v_N(x)| \leq \sqrt{\beta(\tau)}\sigma_N(x) + \gamma(\tau), \forall x \in \mathcal{X}) \geq 1 - \delta \tag{3.36}$$

The uniform error bounds are calculated as the right-hand-side term. Detailed derivation can be found in [141].

3.4.2 Lipschitz Constant Derivation

The Lipschitz constant is calculated as in equation (3.28). Therefore, the partial derivative is the core of computing the Lipschitz constant. The kernel function applied for CNN-GP regression is the squared exponential kernel, as shown in equation (2.58), and the partial derivative of the squared exponential kernel can be derived as follows. The first order derivative of the dK with respect to. x_i is derived as,

$$\frac{dK}{d\mathbf{x}_i} = -\frac{\mathbf{x} - \mathbf{x}'}{l^2}K. \tag{3.37}$$

Notice that $\mathbf{x}$ and $\mathbf{x}'$ are two different features extracted by CNN. Similarly, the derivative of dK with respect to x_i' is as

$$\frac{dK}{d\mathbf{x}_i'} = \frac{\mathbf{x} - \mathbf{x}'}{l^2}K. \tag{3.38}$$

Note that $\frac{dK}{d\mathbf{x}_i}$ and $\frac{dK}{d\mathbf{x}_i'}$ have an opposite sign. The second derivative can be derived as,

$$\begin{aligned} \frac{ddK}{d\mathbf{x}_i d\mathbf{x}_i'} &= \frac{1}{l^2}K - \frac{\mathbf{x} - \mathbf{x}'}{l^2}\frac{dK}{d\mathbf{x}_i'} \\ &= \frac{1}{l^2}K + \frac{\mathbf{x} - \mathbf{x}'}{l^2}\frac{dK}{d\mathbf{x}_i}. \end{aligned} \tag{3.39}$$

To obtain the Lipschitz constant of $L_K^{\partial i}$, x_i is taken as fixed and leave only x_i' varing. Hence

$$\begin{aligned} \frac{dddK}{d\mathbf{x}_i d\mathbf{x}_i' d\mathbf{x}_i'} &= \frac{1}{l^2}\frac{dK}{d\mathbf{x}_i'} - \frac{1}{l^2}\frac{dK}{d\mathbf{x}_i} + \frac{\mathbf{x} - \mathbf{x}'}{l^2}\frac{ddK}{d\mathbf{x}_i d\mathbf{x}_i'} \\ &= -\frac{1}{l^2}\frac{dK}{d\mathbf{x}_i} - \frac{1}{l^2}\frac{dK}{d\mathbf{x}_i} + \frac{\mathbf{x} - \mathbf{x}'}{l^2}\frac{ddK}{d\mathbf{x}_i d\mathbf{x}_i'}. \end{aligned} \tag{3.40}$$

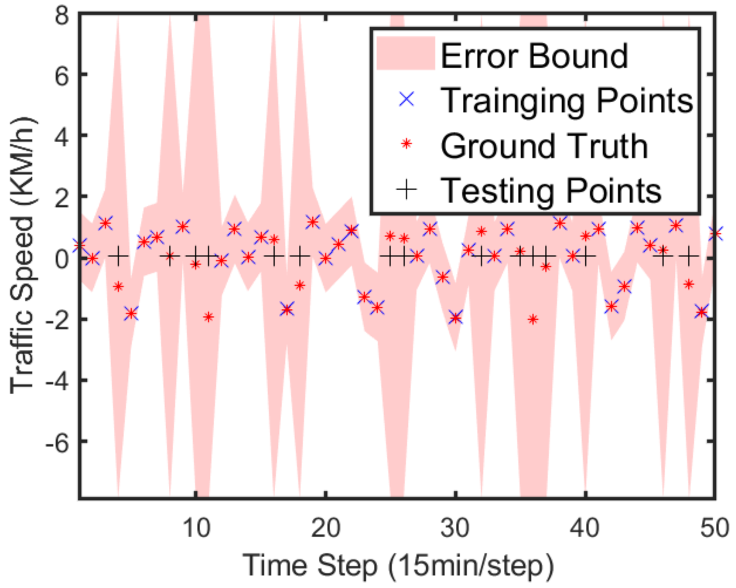


Figure 3.16: *One-step history data of are used to predict one-step head data.* $x = 1 : 1 : 50$, $y = \cos 2x + \sin x + v$.

3.4.3 Evaluation of The Uniform Error Bound with Simulated Data

Before applying the uniform error bound to the real traffic speed data, a simulated time series is used to validate whether the uniform error bound is suitable for time series regression. The GP is employed to fulfil the regression task of a linear combination of sine, cosine functions and noise:

$$y = \cos 2x + \sin x + n, \tag{3.41}$$

where x is the input, y is the output, and $v \sim \mathcal{N}(0, \sigma)$ is the noise. A GP with the kernel of the ARD-SE function and the uniform error bound is applied to perform the regression with uncertainty quantification. The simulated data are divided into 70% for training and 30% for evaluation. First, one-step history data of are used to predict one-step head data, where $x = 1 : 1 : 50$ and $y = \cos 2x + \sin x + v$. As shown in Fig. 3.16, the blue crosses are the training points; red starts are the ground truths; the

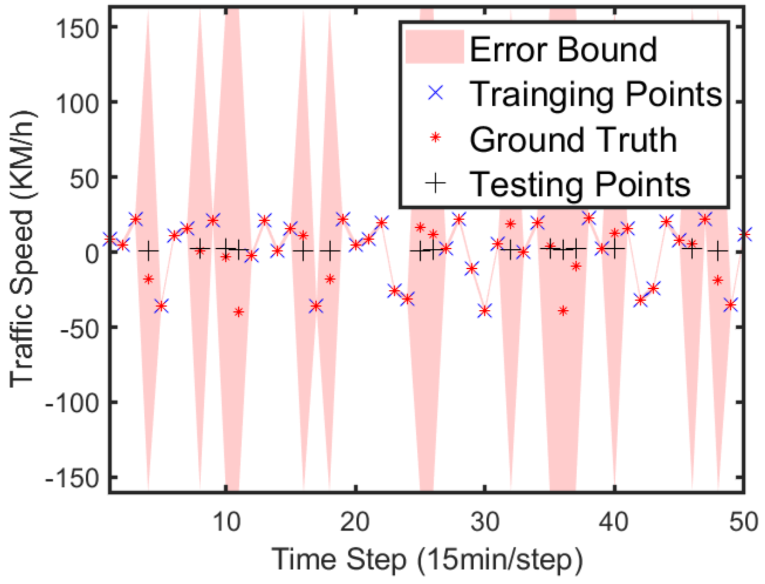


Figure 3.17: *One-step history data of are used to predict one-step head data, where* $x = 1 : 1 : 50$*,* $y = \boldsymbol{20} \times (\cos 2x + \sin x + v)$*.*

black crosses represent the predictions; finally, the pink shades represent the uniform error bounds. There are 50 points in total, and thus 35 points are used for training and 15 points are used for evaluation. The simulation data are more periodic compared with the real traffic speed data. The results show that the performance of the uniform error bound is very poor, as the values of the bounds are extremely large at evaluation points and therefore are unable to quantify the uncertainty well. The first simulated data only contains the periodic characteristic, as the real traffic speed data varies more rapidly. Therefore, the second simulated data is applied for evaluation, which is one-step history data that are used to predict one-step head data, where $x = 1 : 1 : 50$, $y = 20 \times (\cos 2x + \sin x + v)$. With a coefficient, the simulated data changes rapidly, which is more similar as the real traffic speed data. The prediction results are shown in Fig. 3.17. It is even more obvious that the uniform error bounds are acceptable at the training points and are meaningless at the predictions, as the bounds are too wide to quantify the uncertainty at the predictions. This is because the simulated data is too sparse. To prove that the sparsity affects the performance of the uniform error bound,

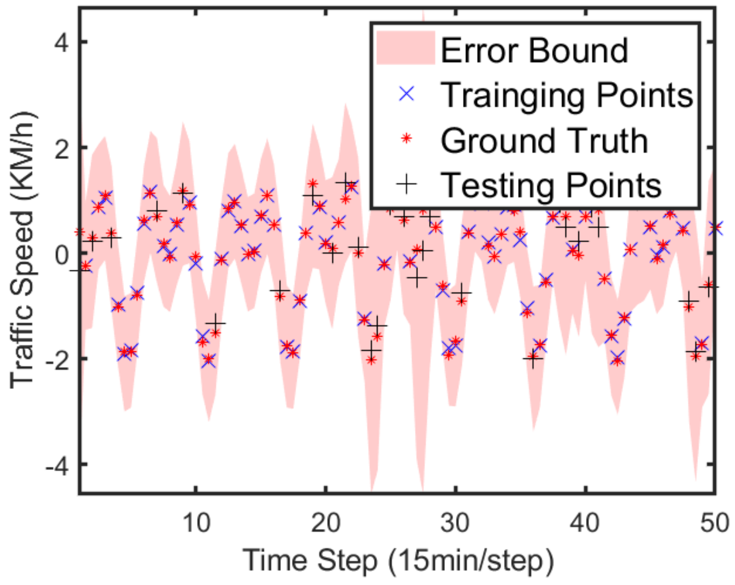


Figure 3.18: *One-step history data of are used to predict one-step head data.* $x = 1 : 0.5 : 50$, $y = \cos 2x + \sin x + v$.

the third and fourth simulated data are applied:

- One-step history data are used to predict one-step head data. $x = 1 : 0.5 : 50$, $y = \cos 2x + \sin x + v$;
- One-step history data are used to predict one-step head data. $x = 1 : 0.5 : 50$, $y = 20 \times (\cos 2x + \sin x + v)$.

Based on the first and second simulated data with 50 points, the numbers of points for the third and fourth simulated data are increased to 100 but remained the value range of the input x the same. As shown in Figs. 3.18 and 3.19, the uniform error bounds are more reasonable compared to the sparse simulated data. More obviously in Fig. 3.19, the uniform error bound near to the training points is narrow, and the points away form training points have large error bounds (eg, $x = 27, 28$), which are reasonable. Fig. 3.20 visualises both the values of the uniform error bound and the values of the prediction variance. The red crosses in the second subplot represent those points as the training points. The relationship between the values of the uniform error bound and

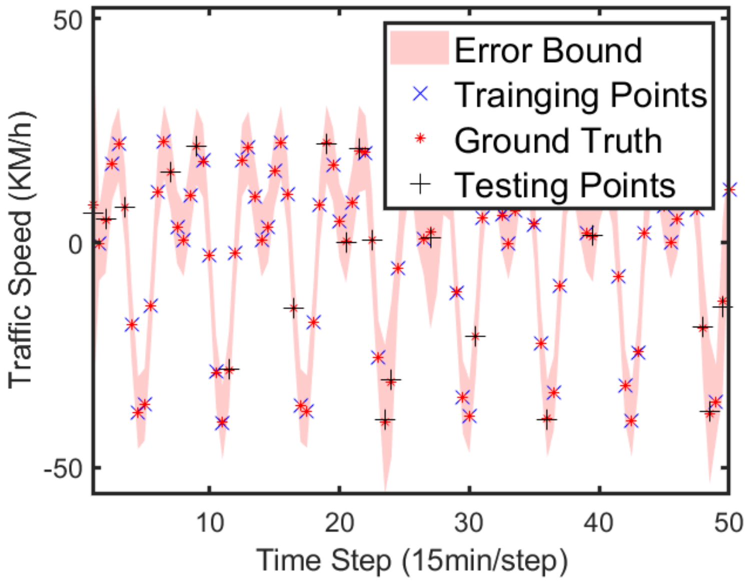


Figure 3.19: *One-step history data of are used to predict one-step head data.* $x = 1 : 0.5 : 50$, $y = \cos 2x + \sin x + v$.

that of the prediction variance can be observed either from visualisation or from the mathematical equation (3.36). The values of the uniform error bound are proportional to the value of prediction variance, since equation (3.36) shows that prediction variance is a coefficient of the bound.

Based on the experimental results, there are three aspects that draw attention. First, the data are sparse, which causes large value of Lipshiz constant, and hence, leads to large uniform error bounds. Second, the prediction variance places huge impact on the error bounds. As shown in equation (3.36), the Lipshiz constant related variable will times the prediction variance, and therefore, if the GP model itself provides large prediction variances, the uniform error bounds will then be larger. Third, as shown in equation (3.34), the increase in the input dimension leads to an exponential increase of M, which finally leads to the increases in the uniform error bounds. The dimension of the data changes the value of beta largely, and thus the number of sensors should be carefully defined.

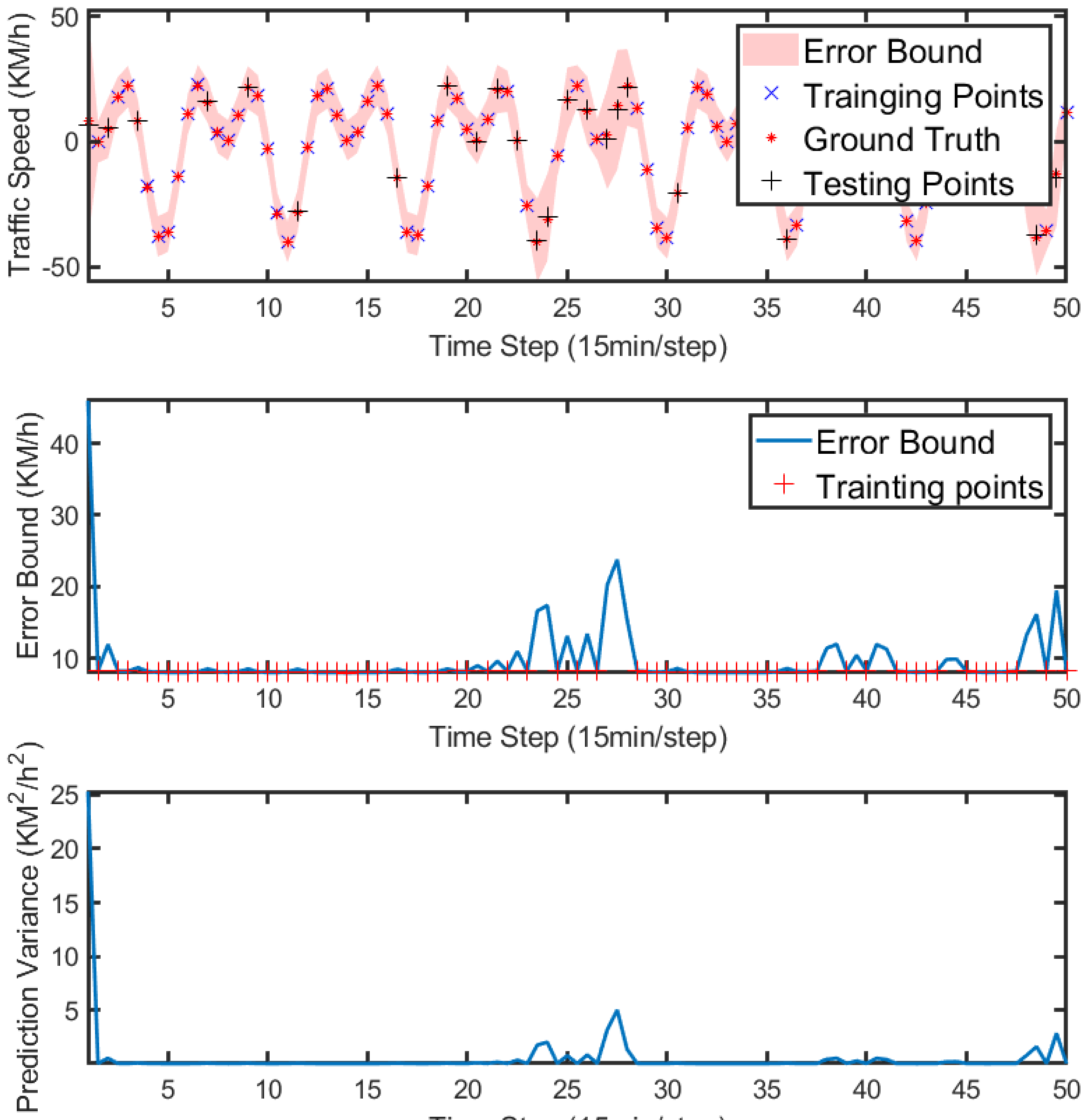


Figure 3.20: *The top figure shows the visualization of UEB for simulated data. The middle figure shows the visualization of UEB value alone the time. The bottom figure shows that prediction variance.*

3.4.4 Evaluation of the Uniform Error Bound with Real Traffic Speed Data

The uniform error bound is applied to quantify the uncertainty and the results of the uniform error bounds are compared with the confidence interval based on the predictive variances. Second, a set of noise with different noise levels is added onto the testing data. The purpose is to observe how the data noise affects the accuracy. Finally, uniform error bounds are applied in both CNN-GP regression framework and long-term GP regression to quantify the uncertainty within the data.

3.4.5 Data Pre-processing

The traffic speed data used are collected on road segments in the city centre of Santander with 15-minute time steps for the year 2016. The traffic dataset is provided thanks to the SETA EU project. The data are divided in the same way as described in Section 3.2.3. The CNN-GP framework with uniform error bound is performed to accomplish the following two tasks:

- Task 1: 1-step ahead prediction on Sensor 1, with 10-step traffic speed history on 10 road segments.
- Task 2: Based on Task 1, different levels of simulated sensor noises are added into the data, and therefore the noisy data becomes $\mathbf{X}_{noisy} = \mathbf{X} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2_{noise})$.

The models are implemented on Python by using Tensorflow, Keras and GPflow. Training and performance evaluation is run on a PC with 8-core i7-10700K CPU, 48 GB memory and an RTX-2080 GPU. The mean squared error (MSE) is used as a loss function when CNN is applied to traffic prediction. The Adam optimizer [104] with exponentially decaying learning rate is utilised to minimise total MSE. The CNN architecture is a 6-layer CNN that contains three pairs of convolutional layers and max

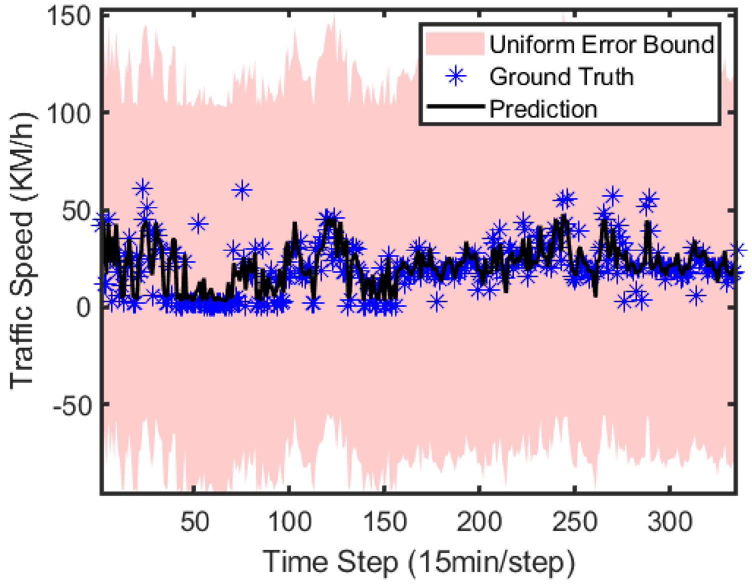


Figure 3.21: *Visualization of UEB for real traffic speed data.*

pooling layers.

$$K_{ARD\_SE}(\mathbf{s},\mathbf{s}') = \sigma_f^2 \exp(-\frac{1}{2l^2}\sum_{j=1}^{q}(\mathbf{v_j}-\mathbf{v_j}')^2), \tag{3.42}$$

Root mean squared error (RMSE) is taken as the benchmark of performance of the traffic speed prediction models. The RMSE averaged over N samples of traffic speed data is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}{N}}, \tag{3.43}$$

where y_i is the ground truth speed value and $\widehat{y_i}$ represents the traffic speed prediction on i-th road segment in the evaluation data set. With a noise-free environment, CNN and CNN-GP regression provide similar RMSE results that are 8.209 km/h and 8.080 km/h respectively. CNN-GP regression provides better performance compared to typical CNN in a noise-free environment. A visualisation of the UEB is shown in Fig. 3.21. By varying the hyperparameter τ, the UEB is minimised in this case. However, the UEB is extremely wide, providing a poor uncertainty quantification. The reason is that the prediction variance itself is large. Recall that the GP regression model is defined as $y = f(x) + \epsilon$ where $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$ [56], and the predictive

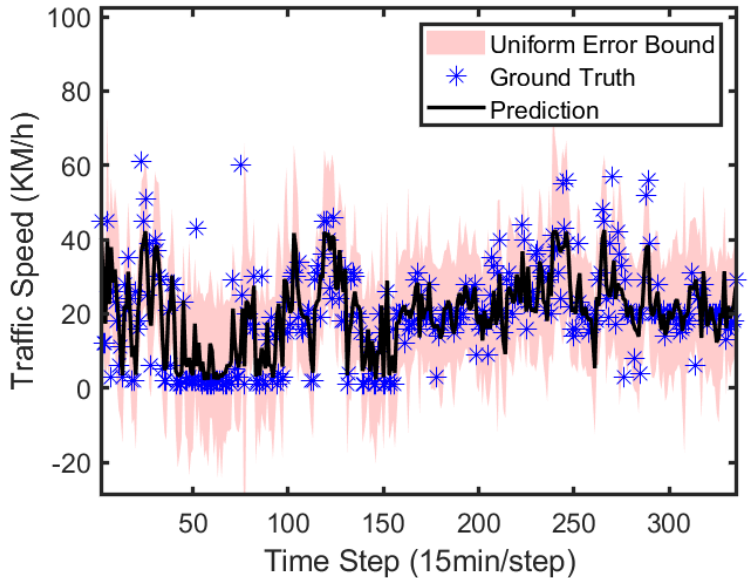


Figure 3.22: *Visualization of UEB for real traffic speed data ($\tau = 1e^{-7}$).*

distribution of the test target is calculated as a summation of the covariance function and $\sigma_n^2\mathbf{I}$. However, in the case of the CNN-GP framework, a GP regression is applied to the features extracted by CNN, and therefore the distribution of the function $f(x)$ is enough to quantify the uncertainty within the traffic speed data.

The results of the UEB for Task 1 is shown in Fig. 3.22. The values of the UEB are now satisfied to quantify the uncertainty. With $\tau = 1e^{-5}$, there are 8 ground truth points (2.39%) outside the UEB; with $\tau = 1e^{-7}$, there are 8 ground truth points (4.78%) outside the UEB. The hypothesis δ in equation (3.36) and δ_L in equation (3.30) are assumed to be 0.1, which means that 90% of the ground truth values are located within the uniform error bounds. Therefore, the hypothesis holds with $\tau = 1e^{-5}$. The widely applied uncertainty quantification method, the confidence interval, shows that 80 ground truth points (23.88%) locate outside the confidence interval with 1-σ confidence intervals, which indicates that the prediction has 68% confidence that the ground truth values are located within the $1 - \sigma$ confidence intervals. When comparing the UEB with the confidence interval, the UEB shows advantages of uncertainty

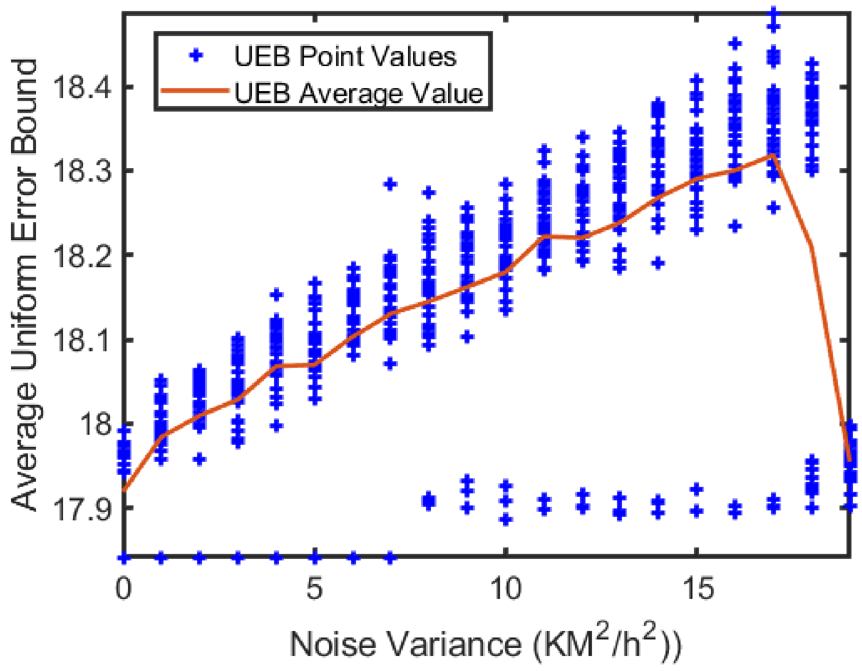


Figure 3.23: *UEB value evolution as the noise variance changes with* $\tau = 1e^{-7}$.

quantification. From the hypothesis view, the UEB provides a more quantified view of the probability that the predictions are located within the intervals. From the statistical point of view, the UEB provides an accuracy error bound that covers most of the ground truth points within the bound, which means that the UEB provides better uncertainty quantification.

For Task 2, different levels of noise are added to the traffic speed data. The variance of the noise varies from 0-19 KM^2/h^2, and 30 experiments are performed at each noise level. In Fig. 3.23, the blue crosses are the average UEB values of each experiment, and the averages at each noise level comprise the red line, which shows that the mean UEB increases progressively as the noise level increases. A turning point exists at the noise level with a variance of 17 KM^2/h^2, and the average UEB starts to decrease. This can be explained by the fact that the noise is strong enough to submerge the traffic speed data.

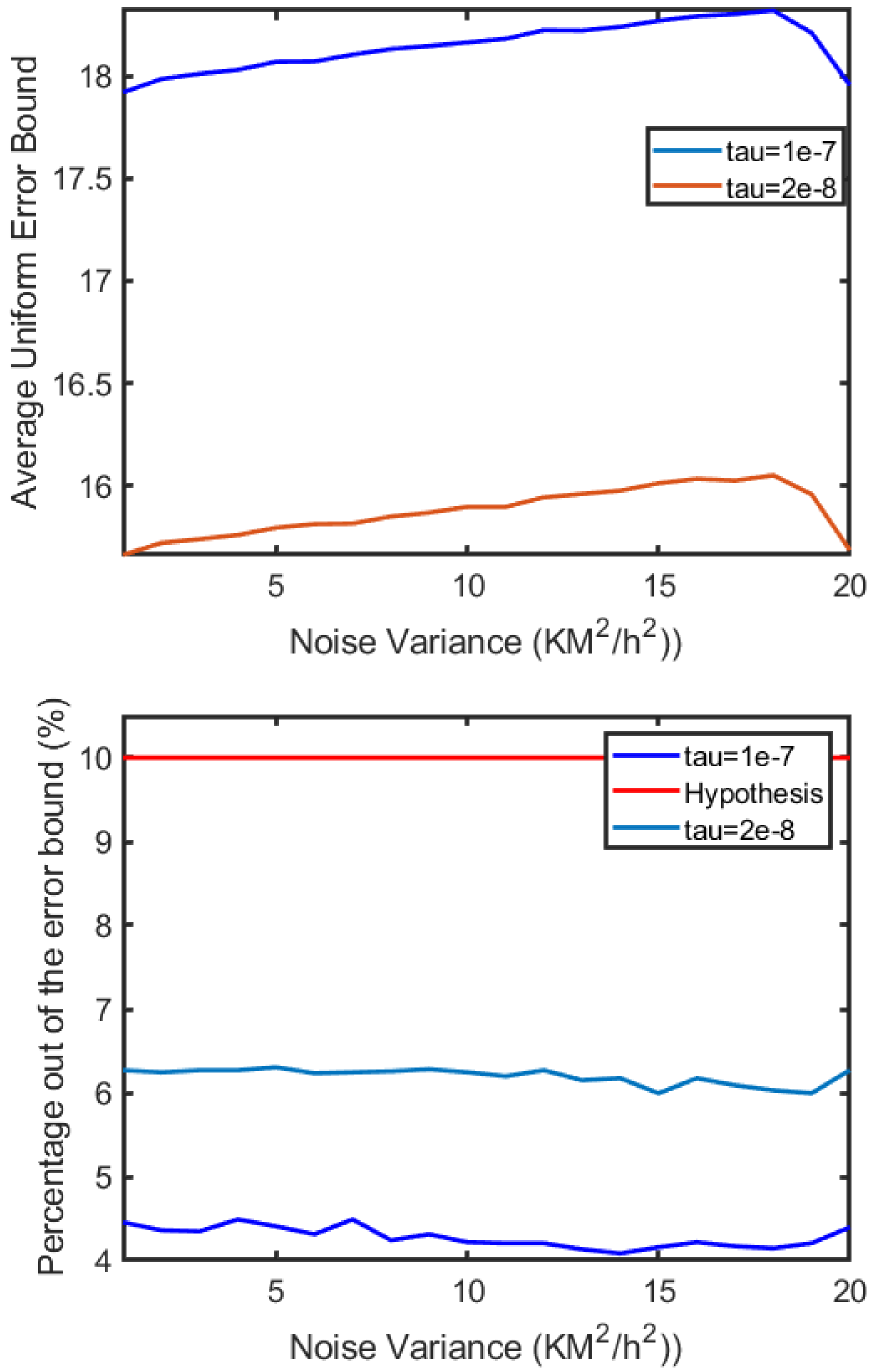


Figure 3.24: *Top: Average uniform error bound value. Bottom: Percentage of the points that is out of the error bounds.*

Fig. 3.24 presents the results of UEB with different values of τ. Generally, the UEB becomes narrower as the value of τ becomes smaller and therefore the percentage of points located outside the UEB increases. In other words, τ is a controller parameter that defines how aggressive uncertainty quantification will be.

3.4.6 Conclusion

In this section, the uniform error bound is introduced for uncertainty quantification. Assumptions are described as they are less restrictive compared with existing error bound methods. In Section 3.4.2, the partial derivatives of CNN-GP with K_{SE} is derived. Before the evaluation with the real traffic data, a simulated data is applied for examine the availability of the UEB for regression problem. Three aspects draw attentions. First, the sparsity of the data will cause a large Lipschitz constant, and therefore, will lead to a large UEB. Second, the UEB is proportional to the prediction variance and, hence, a large prediction variance will cause large UEB. Third, the UEB is exponentially proportional to the dimension of the data. CNN-GP provides a large prediction variance for traffic speed data, and therefore the distribution of the function $f(x)$ for GP is applied, as it is enough to quantify the uncertainty within the traffic speed data. The evaluation of the UEB shows that it is a better uncertainty quantification method compared with the confidence interval. By adding different levels of noises, the UEB values increase as the levels of noise increase, and the UEB still remains the ability to quantify the uncertainty.

3.5 Summary

Numerous methods exist for short-term traffic prediction. However, these approaches often necessitate the development of specific physical models tailored to the particular traffic scenario or solely consider the spatial or temporal relationships between road segments. This chapter first proposes an efficient CapsNet which overcomes the drawbacks of the CNN. The CapsNet replaces the pooling layers with neuron capsules and dynamic routing, and thus it can abstract higher-level features compared with the state-of-the-art CNN and encode the probability of those features that locate in the local region. However, the state-of-the-art CNN and CapsNet are limited by the lack of uncertainty analysis. Therefore, this chapter proposes a ConvNet GP framework to endow neural networks with the capability of uncertainty quantification. The ConvNet GP framework is proposed for traffic speed prediction, where the state-of-the-art CNN is equated to a GP. The performances of the proposed approaches are evaluated with real traffic speed data, and the impact of noise is investigated by adding different levels of noise. The results indicate that the properties of GPs offer advantages in evaluating the uncertainties associated with sensor data in traffic speed prediction. Furthermore, the ConvNet GP demonstrates superior performance in terms of RMSE. The confidence intervals are used to quantify the uncertainty. Additionally, inspired by DKL, a novel deep kernel CNN-GP framework for time series prediction is introduced, leveraging a GP regression approach that utilises the last-layer features of the state-of-the-art CNN as input for making predictions. Building upon the CNN-GP framework, this chapter introduces spatio-temporal kernels, allowing it to consider both time and space. By incorporating an additional periodic kernel, the GP is endowed with the capability to capture the inherent periodic patterns within the data and the ability to make short and long term predictions. The experimental results demonstrate that the deep kernel CNN-GP framework with spatio-temporal kernel achieves more accurate predictions compared to the baseline CNN model in both CO_2 concentration and traffic prediction tasks. This improvement highlights the effectiveness of incorporating the GP module with the spatio-temporal kernel. An efficient UEB is proposed for uncertainty quantification by removing the noise term, since the GP regression in the CNN-GP framework focuses on modelling the distribution of unknown function $f(x)$. The experiments are done on the CNN-GP framework with squared exponential kernel. Simulated data is employed to examine the suitability of a UEB for regression problems. The findings shed light on three key aspects that warrant attention: 1.the sparsity of the data results in a substantial Lipschitz constant, thereby yielding a larger uniform error bound; 2. the uniform error bound exhibits a direct proportionality to the prediction variance; 3. the uniform error bound displays an exponential relationship with respect to the dimensionality of the data. The results obtained suggest that the UEB offers a superior approach for quantifying uncertainty compared to the confidence interval. Through the introduction of varying levels of noise, it is observed that the values of the uniform error bound escalate in tandem with the intensity of noise. Importantly, the uniform error bound consistently maintains its ability to effectively quantify uncertainty.

Chapter 4

Deep Learning for Cancer Bone Segmentation

4.1 Introduction

In the UK, one in four deaths occur due to cancer, and at this stage, the cancer has spread to the bones in more than 40% of patients [142]. Bone disease caused by cancer results in substantial pain, loss of mobility and fractures in patients, as well as increasing the fatality and cost of treatment [143], [144]. Unfortunately, there are no pharmacological treatments to help repair bone disease. A major limitation in the development of bone healing drugs is the lack of reliable approaches to accurately quantify bone lesions. Hence, it is essential to develop an automated approach to accurately diagnose bone disease.

Evans et al. developed *Osteolytica* to measure cancer-induced lesions in mouse tibiae scanned by micro computed tomography (µCT) [145]. *Osteolytica* first dilates the sample bone volume image until the holes on the outer surface are filled. Then a contraction is performed on the dilated volume, which stops when the contracted volume reaches the highest overlapping ratio between itself and the original volume. By subtracting the original volume and the contracted volume, additional areas are obtained as lesion areas, and therefore, the areas and the number of the bone lesion can be calculated [145]. The analysis using *Osteolytica* provides 0.53% average variability, which is 37 times more accurate compared to the ImageJ analysis method from [146],[147].

However, a significant problem with *Osteolytica* is that it recognises cartilage, a normal structure in healthy and diseased bones, as a bone lesion, creating a false positive result. Carilage can be manually excluded, but this creates a major problem when bone lesions connect with the growth plate, as the real bone lesion would also be excluded. Therefore, the objective of this study is to segment bone and cancer-induced bone lesion in three dimensions using Micro Computed Tomography (μCT) dataset. Micro Computed Tomography (μCT) datasets were used by scanning the proximal end of mouse tibiae with or without tumour [148]. The μCT datasets contain 2D transverse slices that can be rendered into a 3D dataset. Each slice has width W and height H. By aggregating N slices, the datasets are represented in 3D tensor with $W \times H \times N$. During the experiments described in later sections, the M9 mouse tibiae dataset with tumour is used to evaluate the approaches. M9 dataset has 1235 slices, and each slice has 1440×1440 resolution, which means that M9 dataset is a tensor with dimension of $1440 \times 1440 \times 1235$.

More specifically, machine learning approaches are applied on the pre-clinical μCT datasets of bones with and without cancer. Two main approaches are proposed: the fast edge detection approach with structured forest, an extension of [81], and deep learning approaches, such as convolutional neural network (CNN) [149], [49], [150], capsule network (CapsNet) [151],[96] and Gaussian process (GP) approaches [6]. Ultimately, the objective of this study is to improve the accuracy of quantifying bone lesions to facilitate reliable pre-clinical testing of new bone-targeted therapies.

More specifically, the objective of this study is to improve the accuracy of quantifying bone lesions to facilitate reliable pre-clinical testing of new bone-targeted therapies. Section 4.2 introduces two main approaches: the fast edge detection approach with structured forest, an extension of [81], and deep learning approaches, including a convolutional neural network (CNN) [149], [49], [150], capsule network (CapsNet) [151],[96] and Gaussian process (GP) approaches [6]. The novetly and significance are the following:

- Machine learning approaches are proposed for bone cancer segmentation, providing a new perspective of dealing with bone cancer segmentation.
- A comparative study is done for evaluating the proposed machine learning approaches.

However, Section 4.2 focuses on bone segmentation, and areas of lesions are difficult to locate due to the limitation of the dataset. Section 4.3 introduces an approach to creating simulated data, and hence the lesion areas are correctly labelled. A generative adversarial network (GAN) is proposed to reconstruct cancer bone back to healthy bone. The novelty and contributions are the following:

- An approach for creating simulated cancer bone dataset with correct labels is proposed.
- An GAN is proposed to reconstruct cancer bone to healthy bone, and therefore the location of lesions is obtained by subtracting reconstruction bone and cancer bone.
- GAN provides a potential solution to the challenge of the lesion areas being extremely similar to the background.

4.2 Machine Learning Frameworks for Cancer Bone Segmentation

In this section, three deep learning and one GP related frameworks are introduced. A CNN with dense predictions is proposed to be the base-line. Considering that the strength of the CNN lies in whole-image classification and that classifying each pixel individually can be computationally expensive, a fully convolutional network (FCN) is proposed. The FCN enables pixelwise prediction with the flexibility to handle inputs of any size and provides outputs of the matching dimensions. However, the pooling layers in CNNs and FCNs can lead to the loss of important information in the data, preventing them from effectively recognising pose, texture, and transformations. In the case of the μCT dataset, the positions of mouse tibiae are not guaranteed to be identical, necessitating the introduction of rotation. Therefore, a CapsNet is proposed for cancer bone segmentation. However, despite the advances made in CNNs, FCNs, and CapsNets, there are still inherent challenges, notably the limitations imposed by computational resources. Furthermore, the analysis of uncertainty information, which has crucial significance in high-risk applications such as clinical applications, has not been extensively explored. In this context, a ConvNet GP is proposed for cancer bone segmentation, aiming to enhance both computational efficiency and accuracy.

4.2.1 Convolutional Neural Network

A CNN with dense predictions is designed to be the base-line. The convolution layers have 256, 128 and 64 channels, respectively, and the filter size of each layer is 3×3. A max pooling layer with a filter size of 2×2 and a stride of 2 follows each convolution layer. All the convolution layers are activated by a Rectified Linear Unit (ReLu) activation function. At the end of the network, there is a fully connected layer. The architecture is presented in Fig. 4.1. The network parameters are presented in Table 4.1.

4.2.2 Fully Convolutional Network for Semantic Segmentation

It has been introduced many times in the previous sections that CNNs achieved impressive advantages in recognition. CNNs are not only developed for whole-image

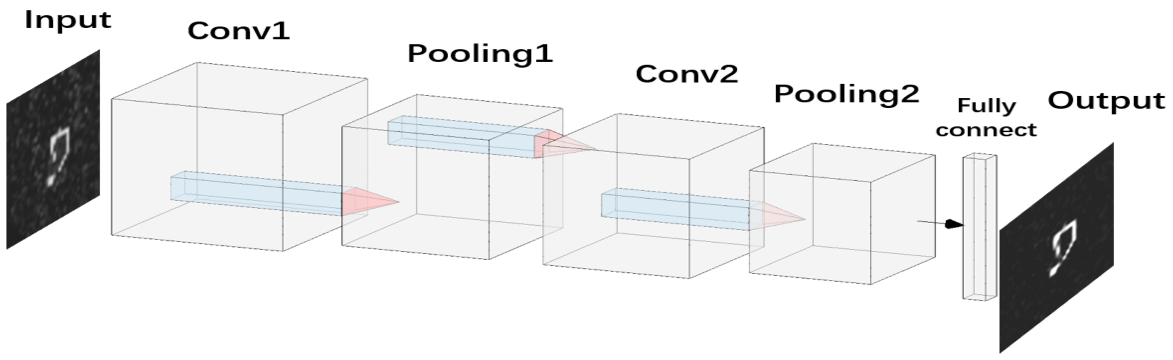


Figure 4.1: *CNN architecture for bone segmentation.*

Table 4.1: *Layer parameters of CNN*

Layer	Parameter	Activation
Convolution1	(256,3,3)	ReLu
Polling1	(2,2)	-
Convolution2	(128,3,3)	ReLu
Polling2	(2,2)	-
Flatterning	-	-
Fully-connected	-	-

Table 4.2: *Layer parameters of FCN*

Layer	Parameter	Activation
Convolution1	(32,3,3)	ReLu
Polling1	(2,2)	-
Convolution2	(32,3,3)	ReLu
Polling2	(2,2)	-
Deconvolution 1	(32,4,4)	ReLu
Deconvolution 1	(32,4,4)	ReLu
Convolution3	(32,2,2)	-
SoftMax	-	-
Pixel Classification	-	-

classification [49, 152] but also for local tasks, including the detection of bounding boxes objects [153, 154, 155] and local correspondence [156, 157]. The basic components, including convolution and pooling, operate in the local regions and only depend on the relative spatial coordinates of the inputs [150]. Let $\mathbf{X}_{i,j}$ be the input vector on a particular layer and $\alpha_{i,j}$ be the output of the layer. The transformation in the particular layer can be represented in function form as following [150],

$$\alpha_{i,j} = f_{k,s}(\mathbf{X}_{i,j}), \tag{4.1}$$

where k and s are the kernel size and the stride, and $f(\cdot)$ is determined by the layer type: convolutional as shown in equation (2.30), max pooling as shown in equation (2.31) and nonlinear activation function as shown in equation (2.32 - 2.39). The functional form maintains with kernel size and stride obeying the transformation rule [150]:

$$f_{k,s} \circ g_{k\prime,s\prime} = (f \circ g)_{k\prime+(k+1)s\prime,ss\prime}. \tag{4.2}$$

A fully convolutional network (FCN) is a network with only layers in this form of nonlinear filter. An FCN can effectively generate predictions for pixelwise tasks such as semantic segmentation, since it can work on an input of any size and provide an output of the matching dimensions. In the rest of this section, we discuss the way to convert classification networks into FCNs that produce coarse output heatmaps. Deconvolution layers are introduced for upsampling.

FCN is adapted classifiers for dense prediction. Typical classification networks, such as AlexNet [11], end with fully connected layers, and these fully connected layers have specific dimensions without spatial coordinates. However, the fully connected layers can be considered as convolutions of kernels that cover the entire input spaces, which casts the classification networks into fully convolutional networks that produce classification maps. The output classification maps make the fully connected network naturally a choice for dense problems, such as semantic segmentation [150]. If the ground truth is available at each output element, the forward and backward passes are straightforward and benefit from the inherent computational efficiency of convolution. However, the output dimensions of the fully convolutional networks are reduced with respect to the stride of the convolution operation. Dense predictions can be generated from these coarse outputs. One way to connect coarse output to dense predictions is **Upsampling with deconvolution.** Upsampling f times with deconvolution is indeed a convolution with a fractional input stride of $1/f$, as long as the factor f is an integer. Since deconvolution simply reverses the forward and backward passes of convolution, upsampling with deconvolution can be performed within the network for end-to-end learning with backpropagation.

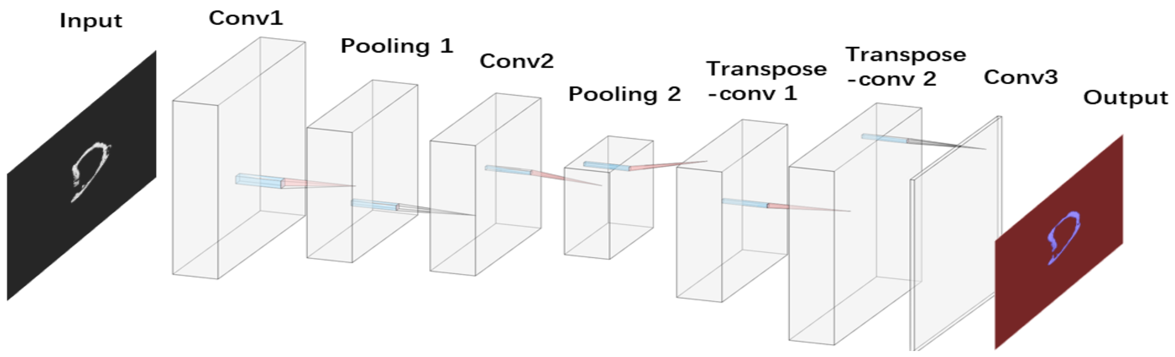


Figure 4.2: *Fully convolutional network architecture for bone segmentation*

Segmentation Architecture is shown in Fig. 4.2. The first four layers are the same as the CNN introduced in Fig. 4.1. These layers performed the same job as in the CNNs, as they extracted deep feature hierarchies that encode the locations and semantics. Therefore, for pixel-wise prediction and classification, the encoded information is required to connect back to pixels. Deconvolutional layers are introduced as an efficient and effective solution. Deconvolution is commonly called backward convolution, which means that deconvolutional layers simply reverse the operations in convolutional layers. Therefore, deconvolutional layers achieve end-to-end learning by backpropagating the pixelwise loss [150]. Different from CNN, FCNs replace fully-connected layers, typically used for classification, by using convolutional layers to classify each pixel in the image.

Table 4.2 lists the parameters of the FCN implemented for this study. Convolutional layers 1 and 2 have 32 filters with 3× 3 filter sizes. The convolutional operations in those two layers are performed with a stride of 1 with 1 padding and activated by a ReLu. Pooling layers have filters of size 2×2 applied with a stride of 2 and 0 padding. 32 filters with 4×4 filter sizes applied with a stride of 1 and 1 cropping are applied in the deconvolutional layers. Convolutional layer 3 has 32 filters of size 2×2 applied with a stride of 1. The pixel classification layer applies cross-entropy as a loss function.

4.2.3 Capsule Network

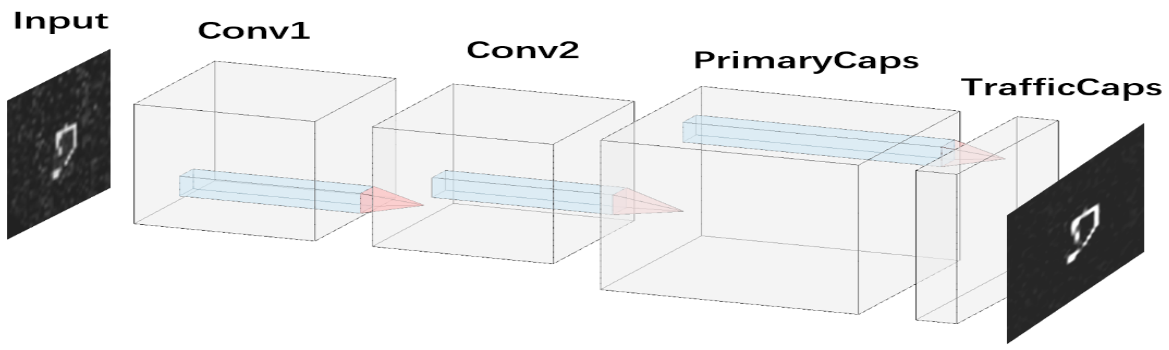


Figure 4.3: *Capsule network architecture for bone segmentation*

CNNs and FCNs have shown a very good performance in different applications. However, max pooling in CNNs and FCNs lose valuable information by selecting the max values in the activations. There are only limited and pre-defined pooling mechanisms to handling variations in the spatial arrangement of data [158]. Since it is impossible to ensure that the positions of the bones are exactly the same while CT scanning, data augmentation or image registration is required for CNNs and FCNs. CapsNet has been proposed in [151] and [96] to address the drawbacks of CNNs and FCNs. Each layer in CapsNet contains capsules that represent different characteristics of the object. The main difference between capsules and artificial neurons is that capsules are in vector forms and their activations provide vector output instead of scalers in artificial neurons. More significantly, the routing algorithm updates the weights between two capsule layers, which determines the way in which low-level capsules feed their input into high-level capsules. Detailed introduction is described in the previous section 3.2.1. The architecture of the CapsNet implemented for this case study is presented in Fig. 4.3.

Convolutional layers 1 and 2 have 32 filters with 3× 3 filter sizes. PrimaryCaps has

128 filters with 3× 3 filter sizes. The convolutional operations in all three layers are performed with a stride of 1 with 0 paddings and activated by a ReLu nonlinear activation. Each capsule in PrimaryCaps is an 8-dimensional vector, and capsules in one cuboid share weights. The last layer is TrafficCpas that has a 16-dimensional capsule per pixel. The routing algorithm performs between PrimaryCpas and TrafficCaps with 3 iterations. The parameters of the proposed CapsNet are listed in Table 4.3.

Table 4.3: *Layer parameters of CapsNet*

Layer	Parameter	Activation
Convolution1	(32,3,3)	ReLu
Convolution2	(32,3,3)	ReLu
PrimaryCaps	(128,3,3)	ReLu
	Capsule zise 8	-
TrafficCaps	Capsule size 16	-

4.2.4 Convolutional Neural Network As Shallow Gaussian Process

Nevertheless, CNNs and FCNs still face challenges, especially since they are time-consuming and computationally expensive. In addition, they provide deterministic results without uncertainty analysis, and therefore uncertainty becomes one of the hidden problems in high-risk applications, such as biomedical applications [159]. A GP approach, one of the most powerful tools in Bayesian inference, has the potential to equip CNNs and FCNs with the capabilities of uncertainty analysis. Garriga et al. [6] proposed that a deep CNN is essentially a shallow GP, which allowed CNN to analyse the uncertainty.

A standard CNN transformation, with L hidden layers, is given as following,

$$a_j^{(l+1)}(X) = b_j^l + \sum_{i=1}^{C^l} W_{j,i}^l \phi(a_i^l(X)), \tag{4.3}$$

where

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{C^0}]^T \tag{4.4}$$

is the input image with height $H^{(0)}$ and width $D^{(0)}$. b_j^l is the bias and $W_{j,i}^l$ is the weight matrix that derives from the filter $U_{j,i}^l$ in the l-th layer. $\phi(a_i^l(X))$ is the activation and $a_i^l(X)$ is the feature map of the previous layer. The elements in a filter $U_{j,i}^l$ are random, and thus the number of potential filters could approach infinity, which means that all filters together should average out the noise and extract the features from a polluted input. As described in [6], every element of $U_{j,i}^l$ is governed by a Gaussian distribution and the bias b_j^l is governed by another Gaussian distribution as shown in equations (4.5) and (4.6) respectively,

$$u_{j,i,x,y}^l \sim \mathcal{N}(0, \frac{\sigma_w^2}{C^l}), \tag{4.5}$$

$$b_j^l \sim \mathcal{N}(0, \sigma_b^2). \tag{4.6}$$

As the weight elements and biases have a Gaussian distribution, the number of the filters, therefore, approaches infinity by sampling from the corresponding Gaussian distribution. The number of filters corresponds to the number of channels in a convolutional layer. With the Central Limit Theorem (CLT), $a_j^{l+1}(X)$ is subjected to a Gaussian distribution as the number of channels approaches infinity.

The element-wise feature map transformation is given as,

$$A_{j,g}^{l+1}(X) = b_j^l + \sum_{i=1}^{C^l} \sum_{h=1}^{H^l D^l} W_{j,i,g,h}^l \phi(A_{i,h}^l(X)), \tag{4.7}$$

where C^l represents the channels. With the equation (4.7), the mean and covariance function can be derived,

$$\mathbb{E}[A^{l+1}_{j,g}(X)] = \mathbb{E}[b^l_j] + \sum_{i=1}^{C^l} \sum_{h=1}^{H^l D^l} \mathbb{E}[W^l_{j,i,g,h} \phi(A^l_{i,h}(X))] = 0 \tag{4.8}$$

$$\begin{aligned} &\mathbb{C}\Bigg[W^l_{j,i,g,h}\phi(A^l_{i,h}(X)), W^l_{j,i',g,h'}\phi(A^l_{i',h'}(X'))\Bigg] \\ &= \sigma_b^2 + \sigma_w^2 \sum_{h \in g\text{th patch}} \mathbb{E}\Big[\phi(A^l_{i,h}(X))\phi(A^l_{i,h}(X'))\Big] \end{aligned} \tag{4.9}$$

In [6], the mean equals to 0. While the covariance function only depends on the expectation of the activation function. According to [6], the activation function is ReLu.

4.2.5 Performance Evaluation

The dataset, M9, is randomly split into 70% training and 30% testing. Due to the limitation of computational resources and efficiency, the slice images were downsampled 53 times from 1440×1440 to 27×27 for the CNN, CapsNet and Convnet GP. Only FCN was still trained with original, 1440×1440, data. The downsampling errors were evaluated using the structural similarity index measure (SSIM) [160]. With 53-time downsampling, the SSIM equals 0.927.

The intersection of union (IOU), given by Equation (4.10), the and root mean square error (RMSE) is defined as Equation (4.11)

$$IOU = \frac{Area\ of\ Overlap}{Area\ of\ Union}, \tag{4.10}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{I}(y_{x,y} - \widehat{y_{x,y}})^2}{N}}. \tag{4.11}$$

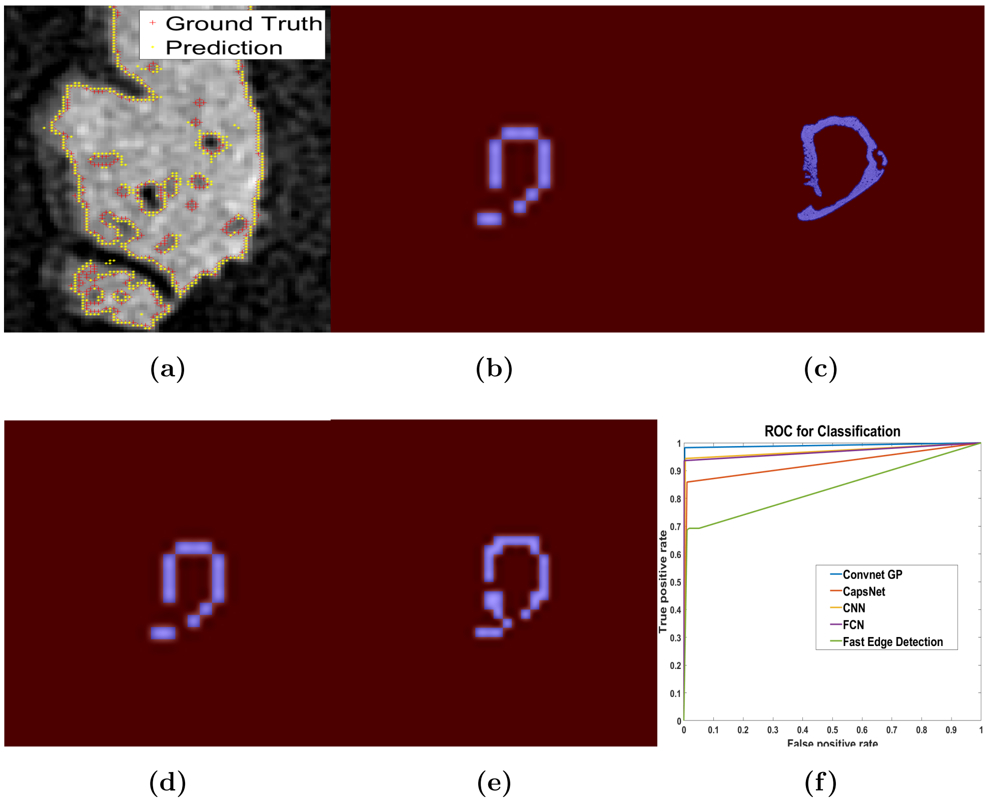


Figure 4.4: *a) Evaluation of the edge detection. Red represents the ground-truth points that are failed to be predicted, and yellow represents the prediction. b) is the visualisation of the result obtained from CNN. c) is the visualisation of the result obtained from FCN. d) Visualisation of the result obtained from CapsNet. e) is the visualisation of the result obtained from Covnet GP. In b), c), d) and e), the red areas represent the background and blue areas represent the bone ares. f) ROC curve.*

are used to evaluate the performance of the approaches. Here $y_{x,y}$ is ground truth pixel value and $\widehat{y_{x,y}}$ is the predicted pixel value.

The result was evaluated by precision and recall that are formulated as following,

$$Precision = \frac{TP}{TP + FP}, \tag{4.12}$$

$$Recall = \frac{TP}{TP + FN}, \tag{4.13}$$

where TP represents true positives, FP represents false positives, and FN represents false negatives. Precision represents the percentage of correct predictions in the total number of predictions. The recall is the fraction of the predictions that are actually correct. Since there was a large number of pixels classified as background rather than edge, to make the result more accurate, only the results within specific bounding boxes that fully enclose the bone are evaluated.

Table 4.4: *Deep learning approach evaluation*

Approach	IOU	RMSE	Percision	Recall
CNN	99.33%	0.065	0.995	0.998
FCN (Full resoultion dataset)	86.12%	-	0.996	0.996
CapsNet	98.58%	0.086	0.989	0.996
Convnet GP	99.60%	0.031	0.997	0.999

CNN and CapsNet were trained with a common starting learning rate of 0.0005 and an exponential decay rate of 0.9999. The FCN was trained with a learning rate of 0.001 and an exponential decay rate of 0.9. The results were evaluated by IOU, RMSE and precision and recall, and their values are given in Table 4.4. The ConvNet GP approach has achieved the best segmentation performance with downsampled data. However, FCN achieves acceptable results with full-resolution data. On the hand of the generalisation of the model, CNN, FCN and ConvNet GP have their drawbacks on handling with rotation, and thus, the data augmentation is required to be applied. On the aspect of computational complexity, the applied CapsNet consumes the longest time, 62 hours, for training. The FCN takes second place with 31 minutes. CNN takes 3.53 seconds, and the ConNnet GP takes 2.15 seconds for training.

4.2.6 Conclusions

This section introduces fast edge detection and four deep learning neural networks to cancer bones segmented from the μ CT datasets. The fast edge detection approach has provided 0.529 precision and 0.574 recall, which are less acceptable compared with deep learning approaches. The other four deep learning NNs have provided outstanding segmentation results with a preclinical dataset. Convnet GP has achieved the highest accuracy respected either on IOU or pixel-wise evaluation (RMSE, precision and recall). However, FCN has the ability to process large-scale data, and CapsNet is rotation invariant. FCN, CapsNet and Convnet GP have different strengths.

This work provides a new perspective of dealing with bone cancer segmentation and compares the effectiveness of machine learning approaches for this challenging segmentation problem. In the next stage of the research, we aim to segment the lesion area from the datasets with artificial lesions with a user-defined size to test the accuracy of our deep learning approaches in three dimensions. A challenge we face is that, while the bone is easily identifiable, the bone lesion areas are almost the same as the background. A limitation in our current approach is that it requires downsampled datasets. Since some information within the dataset is lost during downsampling, the full-size dataset will be processed in the next stage to improve accuracy. Furthermore, the experiments have been performed on 2D slices of the μCT dataset. Given that bone lesions are a 3D structure, it is possible that 2D CNN will not be sufficient to process the dataset. Therefore, the implementation of 3D CNN is a potential architecture that can be investigated in parallel.

4.3 Lesion Bone Reconstruction with Simulated Lesions

Section 4.2 described a comparative study of machine learning methods for bone segmentation problems, which is the first stage of the lesion segmentation task. The next stage aims to segment the lesion areas, which means that the number of lesion areas is able to be calculated, and therefore, the evolution of the lesion can be quantified during the development of bone-healing drugs. However, there are two challenges. First, the size of the datasets is limited, as there are only 6 μCT datasets available, and three of them are with tumours and lesions. Most importantly, the datasets are not labelled, which means that the lesion areas are not labelled. Second, the biggest challenge is that the lesion areas in the μCT slices are identical to either the background or the bone. Based on these challenges, an enhanced dataset is developed and introduced in the next section.

4.3.1 Data Augmentation

Since the machine learning methods have achieved incredible progress in dicrimiative tasks, data augmentation has been one of the most important techniques to improve model benchmarks and solve overfitting problems. There are two main augmentation categories, image manipulations, and deep learning approaches. Image manipulations directly apply to the image data, such as geometric transformation, flipping and kernel filters [97]. Random erasing, proposed by Zhong et al. [161], is one interesting augmentation technique is that designed to create data for recognising occluded images [97]. Random erasing by randomly masking a $n \times m$ patch of an image with certain colour or noise [97]. Inspired by the random erasing technique, a random mask is applied to the μCT dataset without lesion. There are five steps to create simulated datasets with lesions. First, a random noise with the same dimension of μCT slices is generated.

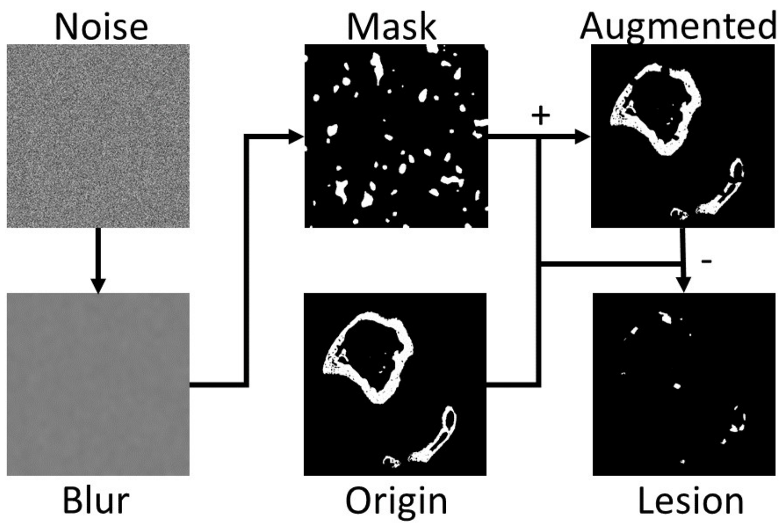


Figure 4.5: *The generation of the simulated data.*

Second, the noise is blurred by Gaussian blur,

$$G(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 - y^2}{2\sigma^2}} \tag{4.14}$$

where x and y are the horizontal and vertical locations in the image respectively, and the standard deviation of the Gaussian distribution is denoted as σ. After the noise is blurred, the pixel values are rescaled from 0 to 255. Third, a mask is created with a certain threshold, with open-and-close techniques in morphology. Fourth, the inverted masks are added to the original μ CT dataset, and therefore the simulated dataset with lesion is generated. Finally, the labels of the lesion can be generated simply by subtracting the original image and the simulated data. The procedures are shown in Fig. 4.5.

4.3.2 Deep Convolutional Generative Adversarial Networks

After developing a bone lesion dataset with labels, the other challenge becomes the biggest obstacle to accomplish the segmentation task. A bone lesion is considered as the abnormal area caused by uncontrollably dividing and multiplying of the cells in the bone, and therefore the lesion areas have the same characteristics as the background

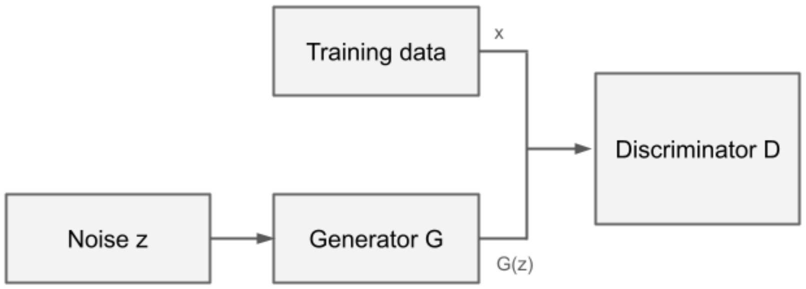


Figure 4.6: *A basic architecture of GAN.*

or the bone in the CT images. In this case, it is even difficult for a not so experienced doctor to recognise the lesions on the CT images. As a result, the machine learning methods proposed in the previous sections are not able to segment the lesion, since the lesion will be either classified into bone or background. Therefore, direct segmentation is too challenging to achieve. However, inspired by the procedures to creating simulated dataset with lesion, the idea of reconstructing the lesion bone back to a healthy one draws the research interest. In this case, the generative adversarial network (GAN) emerges.

The GAN is based on game theory [162]. Two game participants are assumed, in GAN two machine learning models, generator and discriminator. The generator learns the distribution of real data, and on the other hand, the discriminator aims to classify the input to real data and the results generated by the generator [163]. The generator and the discriminator are continuously optimised to, respectively, improve the generation and discrimination abilities for winning the game. The optimisation goal is to achieve a Nash equilibrium between the generator and the discriminator. The basic architecture of GAN is presented in Fig. 4.6.

Generator: generator obeys the real data probability density function, $p_{data}(\mathbf{x})$. However, the generator is not necessary to evaluate the probability density function p_{data}, but the generator draws samples from p_{data}. The generator is defined by a prior distribution $p(\mathbf{z})$ over the input of the generator $\mathbf{z}$, and therefore the generator function is defined as $\mathbf{G}(\mathbf{z};\theta_{\mathbf{G}})$, where $\theta_{\mathbf{G}}$ is a vector of the trainable parameters of the generator [164]. The input of the generator is considered to be the source of randomness, and the prior distribution $p(\mathbf{z})$ should be an unstructured distribution, and the Gaussian distribution is applied in this case. Therefore, the input vector $\mathbf{z}$ is typically a noise. The aim of the generator is to learn $\mathbf{G}(\mathbf{z};\theta_{\mathbf{G}})$ and thus transform the noise $\mathbf{z}$ into the real data [164].

Discriminator: the discriminator plays against the generator. The discriminator examines its input and determines whether the input is drawn from the training distribution or from the generator, which is defined as $\mathbf{D}(\mathbf{x};\theta_{\mathbf{D}})$. The discriminator estimates the probability that the input will be drawn from the training distribution rather than from the generator.

Loss functions: the generator and discriminator have their own loss functions: $\mathcal{L}_{\mathbf{G}}(\theta_{\mathbf{G}},\theta_{\mathbf{D}})$ for the generator and $\mathcal{L}_{\mathbf{D}}(\theta_{\mathbf{G}},\theta_{\mathbf{D}})$ for the discriminator. Briefly speaking, the loss of the generator encourages it to generate outputs $G(\mathbf{z})$ similar to the real data, and the loss of the discriminator encourages the discriminator to correctly classify the input as real or fake [164]. In the case of reconstructing the lesion bone, a deep convolutional GAN (DCGAN) [165] is applied. The DCGAN generator is a deconvolutional neural network, while the discriminator is a CNN. The output of the discriminator is the probability that the input is classified as real data. Therefore, the output values fall in the range of 0 and 1. The input of the discriminator looks more like the real data if the output value is close to 1, and, on the contrary, the input of the discriminator looks more like a fake if the output value is close to 0. To generate multiple classifications, the output of the discriminator is replaced by a softmax function [165], which makes the discriminator the standard classifier for multiclass. Let $\mathbf{z}$ be a random vector with a uniform noise distribution, and the generator function $\mathbf{G}(\mathbf{z})$ maps the random vector to the real data space. Then, assume $\mathbf{x}$ which has a distribution $p_{data}(\mathbf{x},\mathbf{y})$ is the input of the discriminator with the label $\mathbf{y}$. Let k be the number of classes, and let the output of the discriminator be a vector probability of k dimensional $\mathbf{p}$. Then, the loss function of the DCGAN can be formulated as a minimisation problem.

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{data}(\mathbf{x},\mathbf{y})}[\mathbf{L}(\mathbf{y}|\mathbf{x},\mathbf{y} < k)] - \mathbb{E}_{\mathbf{x}\sim\mathbf{G(z)}}[\mathbf{L}(\mathbf{y}|\mathbf{G}(\mathbf{z}), y = k)], \tag{4.15}$$

where $\mathbf{L}$ is the cross-entropy loss function, which is formulated as,

$$\mathbf{L}(\mathbf{y}|\mathbf{x}) = -\sum_{i} y_i' log(p_i). \tag{4.16}$$

In equation (4.16), y' is the expected class, and p_i is the probability that the input belongs to y'. In equation (4.15), $\mathbf{L}(\mathbf{y}|\mathbf{x},\mathbf{y} < k)$ is also the cross-entropy loss function.

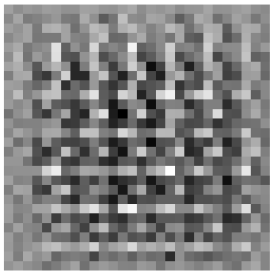
$$\mathbf{L}(\mathbf{y}|\mathbf{x},\mathbf{y} < k) = -\sum_{i}^{k} y_i' log(p_i). \tag{4.17}$$

4.3.2.1 Experiments

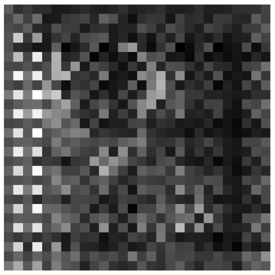
The experiments are performed on the simulated dataset as described in section 4.3. Based on the approach introduced in Section 4.3, 1000 simulated CT slices are created by adding masks on one selected CT slice of the healthy bone. The images are downsampled from 1440×1440 pixels to 28 pixels for computational efficiency. The DCGAN model is implemented in Python using Tensorflow and Keras. The training is run on the Colaboratory from Google with 2 Intel Xeon CPU, 12 GB memory, and a Mostly K80 GPU. The DCGAN model took 56.1 seconds to train one iteration. Two experiments are performed:

- Task 1: Reconstruct 1 2-D CT slice of healthy bone from a noise.
- Task 2: Reconstruct 1 2-D CT slice of healthy bone from a simulated bone with lesions.

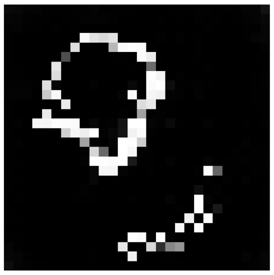
For Task 1, the generator is a deconvolutional neural network that takes a noise vector with a size of 1×100. The kernel size of the deconvolutional layers is 5×5 and the activation function is a non-linear LeakyRelu function. The strides of the deconvolutional layer 1 is 1, and those of deconvolutional layers 2 and 3 are 2. The padding of all



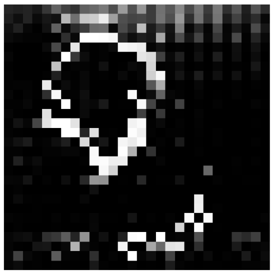
(a) *1*



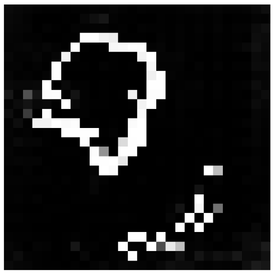
(b) *10*



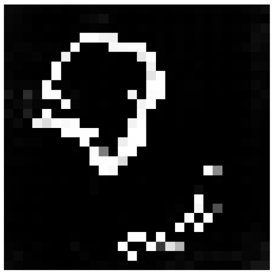
(c) *20*



(d) *30*

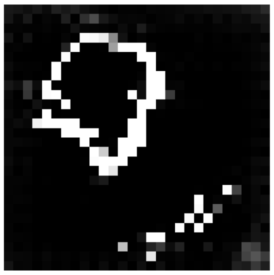


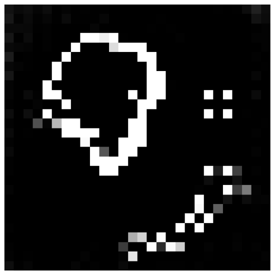
(e) *40*

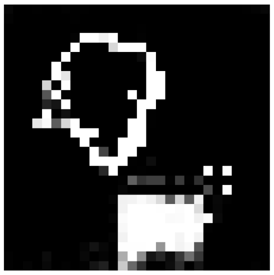


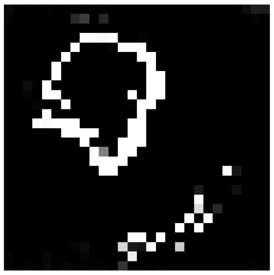
(f) *50*

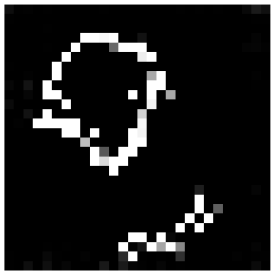
Figure 4.7: *A visualization of GAN generating bone from a noise.*

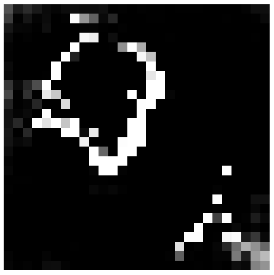

(a) *1*


(b) *10*


(c) *20*


(d) *30*


(e) *40*


(f) *50*

Layer	Parameter	Activation
Convolution 1	(256,3,3)	ReLu
Pooling 1	(2,2)	-
Convolution 2	(128,3,3)	ReLu
Pooling 2	(2,2)	-
Flatten	-	-
Dense	(12544)	-
Reshape	(7,7,256)	-
Deconvolution 3	(128,5,5)	LeakyReLu
Deconvolition 4	(64,5,5)	LeakyReLu
Deconvolution 5	(1,5,5))	LeakyReLu

Figure 4.8: *A visualization of GAN generating bone from cancer bone.*

Table 4.5: *Layer parameters of Generator for reconstructing bone from image with lesion.*

Layer	Parameter	Activation
Dense	(12544)	-
Reshape	(7,7,256)	-
Deconvolution 1	(128,5,5)	LeakyReLu
Deconvolition 2	(64,5,5)	LeakyReLu
Deconvolution 3	(1,5,5))	LeakyReLu

Table 4.6: *Layer parameters of Discriminator*

Layer	Parameter	Activation
Convolution 1	(64,5,5)	LeakyReLu
Dropout 1	(0.3)	-
Convolution 2	(128,5,5)	LeakyReLu
Dropout 1	(0.3)	-
Flatten	-	-
Dense	(28,28)	-

the layers is the same padding. The detailed layer parameters are listed in Table 4.5. The discriminator is a pixel-wise classifier that has two convolutional layers with stride of 2 and is followed by a dropout layer with probability of 0.3. At the end, there is one flatten layer and a fully connected layer. The detailed layer parameters are listed in Table 4.6. The results are shown in Fig. 4.7. The images labelled (a) to (f) are the results generated by the generator from iterations 1 to 50. The DCGAN is trained for 50 iterations in total, and at iteration 50, the DCGAN provides a reconstruction result with 0.9861 IOU (referring to equation (4.10)).

For Task 2, the generator is modified, since the generator input is no longer the noise vector but a CT slice of the bone with lesion. Therefore, two pairs of convolutional layer and a pooling layer and a flatten layer are added at the beginning. The added convolutional layers have kernel size of 3×3 and 256 and 128 filters, respectively. The layer parameters are listed in the bottom table of Fig. 4.8. The discriminator remains unchanged. The results are shown at the top of Fig. 4.8. The DCGAN is trained for 50 iterations in total, and the IOU for the outputs generated by the generator are 0.8533 for iteration 1, 0.8831 for iteration 10, 0.5641 for iteration 520, 0.9315 for iteration

30, 0.8493 for iteration 40 and 0.6395 for iteration 50. Observing the evolution of the training, the best result is provided in iteration 31. The reason why the generation becomes blurred is due to the simulation data. Since the simulation data are created by adding random noise to the CT image of healthy bone and 1000 random noises are added, the DCGAN get overfitted with the random noise as the training iteration goes up.

4.4 Conclusion

In this section, the limitation of deep learning methods for lesion segmentation is introduced, such as the limited number of data and the lack of sufficient labels. Therefore, a data augmentation method is proposed in Section 4.3, so that as many data as possible can be created with the correct labels. Another problem of lesion segmentation is that the lesion areas have identical characteristics as the background or bone in CT images. Therefore, the reconstruction of the lesion bone back to a health bone draws the interest. The GAN is introduced. Two experiments are used to evaluate the performance of the GAN. The first is to use the noise input to reconstruct the healthy bone and the second is to use the lesion bone as the input to reconstruct the healthy bone. The best performances are, respectively, 0.9861 and 0.9315 in terms of IOU. Therefore, it is a possible way to reconstruct the lesion bone back to what a healthy bone should look like, and the area of the lesion can be segmented by subtracting the original and the reconstructed image.

4.5 Summary

This Chapter mainly introduces two stages of cancer bone segmentation. The first stage is to introduce machine learning approaches for bone segmentation. Four types of efficient deep learning approaches are proposed and compared with the base-line methods, fast edge detection, described in Section 2.4. ConvNet GP has achieved the highest accuracy respected either on IOU or pixel-wise evaluation (RMSE, precision and recall). However, FCN has the ability to process large-scale data, and CapsNet is rotation invariant. FCN, CapsNet and Convnet GP have different strengths. The second stage proposes the difficulties of segmenting the area of the lesion, such as the limited number of data and the lack of sufficient labels. Therefore, a data augmentation method is proposed in Section 4.3, so that as many data as possible can be created with the correct labels. Then a generative adversarial network is proposed for the reconstruction of the bone with the lesion. The best performances are 0.9861 and 0.9315 in terms of IOU for inputting noise and lesion image into the generator, respectively. Therefore, it is a possible way to reconstruct the lesion bone back to what a healthy bone should look like, and the area of the lesion can be segmented by subtracting the original and reconstructed image.

Chapter 5

Conclusions and Future Works

The thesis proposes machine learning methods for traffic speed prediction an cancer bone segmentation, and provides methods of uncertainty quantification.

Machine learning methods are proposed for both short-term and long-term traffic speed prediction. Inspired by the CNN for traffic prediction proposed by Ma et al. [45], the traffic speed data is converted into a form of image and, based on this, an efficient CapsNet and a ConvNet GP are proposed and evaluated for the short-term traffic speed prediction. An efficient architecture of CapsNet is proposed for short-term traffic speed prediction, which is the first time applied to the traffic prediction problem. The proposed CapsNet overcomes the disadvantages of CNN and encodes features with the probability of higher-level features located in the local region. A ConvNet GP is proposed for traffic speed prediction by equating the base-line CNN to a GP, providing additional uncertainty information. The evaluation results show that CapsNet has better ability to learn the spatio-temporal features compared with CNN, and the results imply that ConvNetGP has better capabilities on learning in the presence of uncertainties. Different levels of noise are added to the data to evaluate the effects of uncertainty, and confidence intervals are used to quantify the uncertainty. The results show that the ConvNet GP achieves the best performance at all noise levels, and the

RMSE generally increases as the noise level increases for all proposed methods.

Unlike ConvNetGP, a novel deep kernel CNN-GP framework is proposed for time series prediction. The CNN-GP framework combines a CNN module for feature embedding with a GP regression module for prediction and uncertainty quantification. Additionally, a spatio-temporal kernel is proposed for the GP module, allowing it to consider both time and space. Two case studies are conducted to validate and evaluate the performance of the proposed CNN-GP framework. The experimental results demonstrate that the CNN-GP framework with the spatio-temporal kernel achieves more accurate predictions compared to the baseline CNN model in both CO_2 concentration and traffic prediction tasks. This improvement highlights the effectiveness of incorporating the GP module with the spatio-temporal kernel. It enables the framework to generate accurate predictions, even in scenarios where the CNN embedding module may not perform well.

Finally, an efficient uniform error bound is proposed for uncertainty quantification. The experiments are done on the CNN-GP framework with squared exponential kernel. The simulation data are first applied to examine the amiability of uniform error bound for regression problems. Three aspects of the uniform error bound draw attention: the sparsity of the data will cause a large Lipschitz constant, which leads to a large uniform error bound; the uniform error bound is proportional to the prediction variance; the uniform error bound is exponentially proportional to the dimension of the data. As the GP regression takes the last-layer features from the CNN in the CNN-GP framework, the CNN can be considered as the latent function and the GP regression focuses on modelling the latent function itself. Therefore, the noise term in the GP is removed to generate the UEB for CNN-GP. The results imply that the uniform error bound is a better way to quantify the uncertainty compared to the confidence interval. By adding different levels of noises, the values of uniform error bound increase as the levels of noise increase, and the uniform error bound still remains the ability to quantify the uncertainty.

Machine learning methods are proposed for cancer bone segmentation. Two stages are described. The first stage is to introduce machine learning approaches for bone segmentation. A CNN, a CapsNet, a FCN and a ConvNet GP are proposed for bone segmentation and compared with the base-line methods, fast edge detection. These machine learning methods have their strengths. ConvNet GP has achieved the highest accuracy either on IOU or pixel-wise evaluation (RMSE, precision and recall); the FCN has the ability to process large-scale data; and the CapsNet is rotation invariant. The second stage proposes the difficulties of segmenting the lesion area, such as the limited number of data and the lack of sufficient labels. Therefore, a data augmentation method is proposed, so that as many data can be created with correct labels. Then a GAN is proposed for the reconstruction of the lesion bone. The best IOU results are 0.9861 and 0.9315, respectively, for the cases of bone reconstruction from a noisy vector and from a bone image with a lesion image. Therefore, it is a possible way to reconstruct the lesion bone back to what a healthy bone should look like, and the area of the lesion can be segmented by subtracting the original and reconstructed image.

5.1 Future Work

This thesis proposes machine learning methods for both traffic speed prediction and cancer bone segmentation. Below are recommendations for further research. First, the recommendations for traffic speed prediction are given below:

- The uniform error bound is applied on the CNN-GP with the squared exponential kernel. Therefore, the uniform error can be applied to more complex kernel functions, such as the spatio-temporal kernels proposed in Section 3.3. Even more complex, it can be applied on the ConvNet GP, as a CNN can be equivalent to a GP with the central limited theorem.
- Traffic speed prediction is formulated as a regression problem and, therefore, the proposed methods have the potential to solve a high-dimensional regression prob-

lem. The CNN in the CNN-GP framework provides powerful feature extraction abilities, and the GP provides uncertainty quantification.

- Future work will focus on distributed methods for traffic prediction and other ways of uncertainty quantification.

The recommendations for cancer bone segmentation are given below:

- The experiments have been performed on 2D slices of the μCT dataset. Given that the bone lesions are a 3D structure, it is possible that 2D CNN will not be sufficient to process the dataset. Therefore, the implementation of 3D CNN, such as U-Net, is a potential architecture.
- The proposed methods are based on the downsampled data, which is a limitation. Therefore, computationally efficient methods should be investigated to achieve full-size data segmentation and reconstruction.
- Reconstruction methods can be further investigated, as they have shown potential to solve the difficulties that the lesion areas are identical to the background and bone.
- Image inpainting with deep learning, such as [166], can be another recommended direction. As described in *Osteolytica* [145], the arbitrary areas of the lesion can be segmented and thus, with pre-trained image inpainting methods, the lesion areas can be filled. The accurate number of pixels for the lesion can be calculated by subtracting the original and generated images.

This thesis shows that machine learning methods can be effectively employed to solve both regression and classification problems, and the GP can equip neural networks with the ability to characterise uncertainty. The thesis also shows the promising futures of proposed machine learning methods, since uncertainty quantification not only provides higher accuracy, but also reinforces critical decisions. The future work described above for the proposed methods will strengthen their robustness in challenging practical implementations.

Bibliography

[1] R. Qiu and M. Wicks, *Cognitive Networked Sensing and Big Data.* Springer, Boston, USA, 2014.

[2] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[3] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

[4] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, “Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3993–4005, 2015.

[5] Y. Kim, P. Wang, and L. Mihaylova, “Structural recurrent neural network for traffic speed prediction,” in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 5207–5211.

[6] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison, “Deep convolutional networks as shallow Gaussian processes,” in *Proc. of the International Conference on Learning Representations.* Apollo - University of Cambridge Repository, 2019.

[7] L. Rokach and O. Maimon, "Decision trees," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2005, pp. 165–192.

[8] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[9] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[10] M. Van Gerven and E. Bohte, Sander, *Artificial Neural Networks as Models of Neural Information Processing.* Lauzanne: Frontiers in Computational Neuroscience, 2018.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[12] R. S. Sutton, "John mccarthy's definition of intelligence," *Journal of Artificial General Intelligence*, vol. 11, no. 2, pp. 66–67, 2020.

[13] T. M. Mitchell, *Machine learning.* McGraw-Hill, Maidenhead, UK, March 1997.

[14] S. J. Russell and P. Norvig, *Artificial intelligence. A modern approach, 3rd Edition.* Prentice Hall, Pearson Education, Inc., Upper Saddle River, New Jersey, 2010.

[15] H. B. Barlow, "Unsupervised learning," *Neural Computation*, vol. 1, no. 3, pp. 295–311, 1989.

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without

human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[18] S. B. Maind, P. Wankar *et al.*, "Research paper on basic of artificial neural network," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 1, pp. 96–100, 2014.

[19] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. of the 2017 International Conference on Engineering and Technology (ICET)*. IEEE, 2017, pp. 1–6.

[20] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura, "Traffic state estimation on highway: A comprehensive survey," *Annual Reviews in Control*, vol. 43, pp. 128–151, 2017.

[21] P. Wang, Y. Kim, L. Vaci, H. Yang, and L. Mihaylova, "Short-term traffic prediction with vicinity Gaussian process in the presence of missing data," in *Proc. of 2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2018, pp. 1–6.

[22] L. A. Pipes, "An operational analysis of traffic dynamics," *Journal of Applied Physics*, vol. 24, no. 3, pp. 274–281, 1953.

[23] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[24] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[25] A. Gning, L. Mihaylova, and R. K. Boel, "Interval macroscopic models for traffic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 523–536, 2011.

[26] Y. Song, X. Wang, G. Wright, D. Thatcher, P. Wu, and P. Felix, "Traffic volume prediction with segment-based regression kriging and its implementation in assessing the impact of heavy vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 232–243, 2018.

[27] A. C. Atkinson, M. Riani, and A. Corbellini, "The Box–Cox transformation: Review and extensions," *Statistical Science*, vol. 36, no. 2, pp. 239–255, 2021.

[28] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using Box-Jenkins techniques," *Transportation Research Board*, vol. 722, pp. 1–9, 1979.

[29] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[30] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1. AAAI Press, 2017, pp. 1655–1661.

[31] X. Ma, H. Yu, Y. Wang, and Y. Wang, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PloS one*, vol. 10, no. 3, p. e0119044, 2015.

[32] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 166–180, 2018.

[33] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.

[34] J. Park, D. Li, Y. L. Murphey, J. Kristinsson, R. McGee, M. Kuang, and T. Phillips, "Real time vehicle speed prediction using a neural network traffic model," in *Proc. of the 2011 International Joint Conference on Neural Networks*. IEEE, 2011, pp. 2991–2996.

[35] W. Zheng, D.-H. Lee, and Q. Shi, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *Journal of Transportation Engineering*, vol. 132, no. 2, pp. 114–121, 2006.

[36] V. Petridis, A. Kehagias, L. Petrou, A. Bakirtzis, S. Kiartzis, H. Panagiotou, and N. Maslaris, "A Bayesian multiple models combination method for time series prediction," *Journal of Intelligent and Robotic Systems*, vol. 31, no. 1, pp. 69–89, 2001.

[37] S. Ishak, P. Kotha, and C. Alecsandru, "Optimization of dynamic neural network performance for short-term traffic prediction," *Transportation Research Record*, vol. 1836, no. 1, pp. 45–56, 2003.

[38] H. Liu, H. Van Zuylen, H. Van Lint, and M. Salomons, "Predicting urban arterial travel time with state-space neural networks and Kalman filters," *Transportation Research Record*, vol. 1968, no. 1, pp. 99–108, 2006.

[39] L. Shen, "Freeway travel time estimation and prediction using dynamic neural networks," Ph.D. dissertation, Florida International University, USA, 2008.

[40] N. Polson and V. Sokolov, "Deep learning predictors for traffic flows," *arXiv preprint arXiv:1604.04527*, 2016.

[41] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.

[42] F. Gers, "Long short-term memory in recurrent neural networks," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2001.

[43] M. Boden, "A guide to recurrent neural networks and backpropagation," *The Dallas Project*, vol. 2, no. 2, pp. 1–10, 2002.

[44] H. Jaeger, "Long short-term memory in echo state networks: Details of a simulation study," Jacobs University Bremen, Germany, Tech. Rep., 2012.

[45] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[46] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[47] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[48] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of the Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[50] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5308–5317.

[51] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[52] Y. Zhang and Y. Liu, "Traffic forecasting using least squares support vector machines," *Transportmetrica*, vol. 5, no. 3, pp. 193–213, 2009.

[53] J. A. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, “Weighted least squares support vector machines: robustness and sparse approximation,” *Neurocomputing*, vol. 48, no. 1-4, pp. 85–105, 2002.

[54] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, “Support vector regression machines,” in *Proc. of Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9. MIT Press, 1996.

[55] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, “Deep neural networks as Gaussian processes,” *arXiv preprint arXiv:1711.00165*, 2017.

[56] C. K. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2005.

[57] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, “When Gaussian process meets big data: A review of scalable GPs,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[58] A. Malinin, “Uncertainty estimation in deep learning with application to spoken language assessment,” Ph.D. dissertation, University of Cambridge, UK, 2019.

[59] H. Jiang, B. Kim, M. Y. Guan, and M. Gupta, “To trust or not to trust a classifier,” *arXiv preprint arXiv:1805.11783*, pp. 5541–5552, 2018.

[60] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, “A review of uncertainty quantification in deep learning: Techniques, applications and challenges,” *Information Fusion*, vol. 76, no. C, pp. 243–297, December 2021.

[61] E. Begoli, T. Bhattacharya, and D. Kusnezov, “The need for uncertainty quantification in machine-assisted medical decision making,” *Nature Machine Intelligence*, vol. 1, no. 1, pp. 20–23, 2019.

[62] B. T. Phan, "Bayesian deep learning and uncertainty in computer vision," Master's thesis, University of Waterloo, Canada, 2019.

[63] J. Mukhoti and Y. Gal, "Evaluating Bayesian deep learning methods for semantic segmentation," *arXiv preprint arXiv:1811.12709*, 2018.

[64] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, *A Modern Introduction to Probability and Statistics: Understanding Why and How.* Springer, 2005, vol. 488.

[65] L. Laurencelle and F.-A. Dupuis, *Statistical tables, explained and applied.* World Scientific, USA, 2002.

[66] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[67] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos, "The variational approximation for Bayesian inference," *IEEE Signal Processing Magazine*, vol. 25, no. 6, pp. 131–146, 2008.

[68] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[69] J. R. Fram and E. S. Deutsch, "On the quantitative evaluation of edge detection schemes and their comparison with human performance," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 616–628, 1975.

[70] G. S. Robinson, "Color edge detection," *Optical Engineering*, vol. 16, no. 5, pp. 479–484, 1977.

[71] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[72] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[73] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[74] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[75] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[76] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *Proc. of Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 584–592.

[77] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 564–571.

[78] J. J. Lim, P. Dollar, and C. L. Zitnick III, "Learned mid-level representation for contour and object detection," Sep. 18 2014, US Patent App. 13/794,857.

[79] X. Ren, C. C. Fowlkes, and J. Malik, "Figure/ground assignment in natural images," in *Proc. of the European Conference on Computer Vision.* Springer,Boston,, 2006, pp. 614–627.

[80] S. Nowozin and C. H. Lampert, *Structured Learning and Prediction in Computer Vision*, 2011, vol. 6, no. 3-4, pp. 185–365.

[81] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2014.

[82] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: A large margin approach," in *Proc. of the 22nd International Conference on Machine Learning*, 2005, pp. 896–903.

[83] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in *Proc. of the European Conference on Computer Vision.* Springer, 2008, pp. 2–15.

[84] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. of the Twenty-first International Conference on Machine Learning*, 2004, p. 104.

[85] P. Kontschieder, S. R. Bulo, H. Bischof, and M. Pelillo, "Structured class-labels in random forests for semantic image labelling," in *Proc. of the 2011 International Conference on Computer Vision.* IEEE, 2011, pp. 2190–2197.

[86] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[87] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[88] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends® in Computer Ggraphics and Vision*, vol. 7, no. 2–3, pp. 81–227, 2012.

[89] W.-Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery Journal*, vol. 1, no. 1, pp. 14–23, 2011.

[90] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[91] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[92] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proc. of the European Conference on Computer Vision.* Springer, 2012, pp. 746–760.

[93] X. Ren, "Multi-scale improves boundary detection in natural images," in *Proc. of the European Conference on Computer Vision.* Springer, 2008, pp. 533–545.

[94] G. Hinton, "What is wrong with convolutional neural nets?" 06 2018, speech recorded on youtube. [Online]. Available: https://www.youtube.com/watch?v=rTawFwUvnLE

[95] L. A. Dombetzki, "An overview over capsule networks," in *Proc. of Network Architectures and Services*, 2018, pp. 89–95.

[96] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *ArXiv*, pp. 3856–3866, 2017.

[97] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Jata*, vol. 6, no. 1, pp. 1–48, 2019.

[98] M. K. Patrick, A. F. Adekoya, A. A. Mighty, and B. Y. Edward, "Capsule networks–a survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 1, pp. 1295–1310, 2022.

[99] A. Shahroudnejad, P. Afshar, K. N. Plataniotis, and A. Mohammadi, "Improved explainability of capsule networks: Relevance path by agreement," in *Proc. of the IEEE Global Conference on Signal and Information Processing (GLOBASIP).* IEEE, 2018, pp. 549–553.

[100] J. Su, D. V. Vargas, and K. Sakurai, "Attacking convolutional neural network using differential evolution," *IPSJ Transactions on Computer Vision and Applications*, vol. 11, no. 1, pp. 1–16, 2019.

[101] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. of the International Conference on Artificial Neural Networks.* Springer, 2011, pp. 44–51.

[102] J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," *ArXiv*, vol. abs/1711.00165, 2018.

[103] A. Borovykh, "A Gaussian process perspective on convolutional neural networks," *ArXiv*, vol. abs/1810.10798, 2018.

[104] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[105] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 2265–2273.

[106] N. Dereli and M. Saraclar, "Convolutional neural networks for financial text regression," in *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, 2019, pp. 331–337.

[107] Z. Cui, M. Zhang, and Y. Chen, "Deep embedding logistic regression," in *Proc. of 2018 IEEE International Conference on Big Knowledge (ICBK).* IEEE, 2018, pp. 176–183.

[108] W.-J. Chen, J.-J. Yao, and Y.-H. Shao, "Volatility forecasting using deep neural network with time-series feature embedding," *Economic Research-Ekonomska Istraživanja*, vol. 36, no. 1, pp. 1377–1401, 2023.

[109] V. Dondeti, J. D. Bodapati, S. N. Shareef, and N. Veeranjaneyulu, "Deep convolution features in non-linear embedding space for fundus image classification." *Revue d'Intelligence Artificielle*, vol. 34, no. 3, pp. 307–313, 2020.

[110] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis, "Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons," *Journal of Computational Physics*, vol. 477, p. 111902, 2023.

[111] K. Brach, B. Sick, and O. Dürr, "Single shot MC dropout approximation," *arXiv preprint arXiv:2007.03293*, 2020.

[112] M. A. Kupinski, J. W. Hoppin, E. Clarkson, and H. H. Barrett, "Ideal-observer computation in medical imaging with use of Markov-chain Monte Carlo techniques," *Journal of the Optical Society of America A*, vol. 20, no. 3, pp. 430–438, 2003.

[113] J. Swiatkowski, K. Roth, B. Veeling, L. Tran, J. Dillon, J. Snoek, S. Mandt, T. Salimans, R. Jenatton, and S. Nowozin, "The k-tied normal distribution: A compact parameterization of Gaussian mean field posteriors in Bayesian neural networks," in *Proc. of International Conference on Machine Learning.* PMLR, 2020, pp. 9289–9299.

[114] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting.* Springer International Publishing, 2016.

[115] G. Zhang, B. Eddy Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[116] Y. Wang, C. Jing, W. Huang, S. Jin, and X. Lv, “Adaptive spatiotemporal InceptionNet for traffic flow forecasting,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3882–3907, 2023.

[117] G. Huo, Y. Zhang, B. Wang, J. Gao, Y. Hu, and B. Yin, “Hierarchical spatio–temporal graph convolutional networks and transformer network for traffic flow forecasting,” *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[118] D. Cai, X. He, and J. Han, “Spectral regression: A unified subspace learning framework for content-based image retrieval,” in *Proc. of the 15th ACM International Conference on Multimedia*, 2007, pp. 403–412.

[119] C. Hou, F. Nie, D. Yi, and D. Tao, “Discriminative embedded clustering: A framework for grouping high-dimensional data,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 6, pp. 1287–1299, 2014.

[120] J. Ye, R. Janardan, and Q. Li, “Two-dimensional linear discriminant analysis,” in *Proc. of Advances in Neural Information Processing Systems*, 2004.

[121] T. V. Bandos, L. Bruzzone, and G. Camps-Valls, “Classification of hyperspectral images with regularized linear discriminant analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3, pp. 862–873, 2009.

[122] G. Licciardi, P. R. Marpu, J. Chanussot, and J. A. Benediktsson, “Linear versus nonlinear pca for the classification of hyperspectral data based on the extended morphological profiles,” *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 3, pp. 447–451, 2011.

[123] A. Villa, J. A. Benediktsson, J. Chanussot, and C. Jutten, “Hyperspectral image classification with independent component discriminant analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 12, pp. 4865–4876, 2011.

[124] L. M. Bruce, C. H. Koger, and J. Li, “Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction,” *IEEE Transactions on*

Geoscience and Remote Sensing, vol. 40, no. 10, pp. 2331–2338, 2002.

[125] L. O. Jimenez and D. A. Landgrebe, "Hyperspectral data analysis and supervised feature reduction via projection pursuit," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 6, pp. 2653–2667, 1999.

[126] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[127] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in aAplied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[128] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2381–2392, 2015.

[129] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001.

[130] Y. Xie, K. Zhao, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transportation Research Record*, vol. 2165, no. 1, pp. 69–78, 2010.

[131] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Proc. of the 19th International Conference on Artificial Intelligence and Statistics*, vol. 51. PMLR, 2016, pp. 370–378.

[132] S. W. Ober, C. E. Rasmussen, and M. van der Wilk, "The promises and pitfalls of deep kernel learning," in *Proc. of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, vol. 161. PMLR, 2021, pp. 1206–1216.

[133] K. Fang, D. Kifer, K. Lawson, and C. Shen, "Evaluating the potential and challenges of an uncertainty quantification method for long short-term memory mod-

els for soil moisture predictions," *Water Resources Research*, vol. 56, no. 12, p. e2020WR028095, 2020.

[134] S. Sarkka, A. Solin, and J. Hartikainen, "Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 51–61, 2013.

[135] W. Aftab, R. Hostettler, A. De Freitas, M. Arvaneh, and L. Mihaylova, "Spatio-temporal Gaussian process models for extended and group object tracking with irregular shapes," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2137–2151, 2019.

[136] S. Borgeaud, "Gaussian process classifier for CNN uncertainty," 2018. [Online]. Available: https://github.com/seb5666/cnn_gaussian_process_uncertainty/blob/master/report/report.pdf

[137] P. Tans and R. Keeling, "Trends in atmospheric carbon dioxide, carbon cycle greenhouse gases, global monitoring laboratory." [Online]. Available: https://gml.noaa.gov/ccgg/trends/

[138] C. D. Keeling, R. B. Bacastow, A. E. Bainbridge, C. A. Ekdahl Jr, P. R. Guenther, L. S. Waterman, and J. F. Chin, "Atmospheric carbon dioxide variations at Mauna Loa observatory, Hawaii," *Tellus*, vol. 28, no. 6, pp. 538–551, 1976.

[139] "EU SETA project, a ubiquitous data and service ecosystem for better metropolitan mobility, Horizon 2020 Programme, 2016." [Online]. Available: https://figshare.shef.ac.uk/articles/dataset/Traffic_speed_data_for_Santander_city/20164538

[140] X. Liu, L. Mihaylova, J. George, and T. Pham, "Gaussian process upper confidence bounds in distributed point target tracking over wireless sensor networks,"

IEEE Journal of Selected Topics in Signal Processing, vol. 17, no. 1, pp. 295–310, 2023.

[141] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error bounds for Gaussian process regression with application to safe control," vol. 32, 2019.

[142] J. Budczies, M. von Winterfeld, F. Klauschen, M. Bockmayr, J. K. Lennerz, C. Denkert, T. Wolf, A. Warth, M. Dietel, I. Anagnostopoulos *et al.*, "The landscape of metastatic progression patterns across major human cancers," *Oncotarget*, vol. 6, no. 1, p. 570, 2015.

[143] F. Saad, A. Lipton, R. Cook, Y.-M. Chen, M. Smith, and R. Coleman, "Pathologic fractures correlate with reduced survival in patients with malignant bone disease," *Cancer*, vol. 110, no. 8, pp. 1860–1867, 2007.

[144] A. Barlev, "Payer costs for inpatient treatment of pathologic fracture, surgery to bone, and spinal cord compression among patients with multiple myeloma or bone metastasis secondary to prostate or breast cancer," *Journal of Managed Care Pharmacy*, vol. 16, no. 9, pp. 693–702, 2010.

[145] H. Evans, T. Karmakharm, M. Lawson, R. Walker, W. Harris, C. Fellows, I. Huggins, P. Richmond, and A. Chantry, "Osteolytica: An automated image analysis software package that rapidly measures cancer-induced osteolytic lesions in in vivo models with greater reproducibility compared to other commonly used methods," *Bone*, vol. 83, pp. 9–16, 2016.

[146] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "Nih image to imagej: 25 years of image analysis," *Nature Methods*, vol. 9, no. 7, pp. 671–675, 2012.

[147] M. D. Abràmoff, P. J. Magalhães, and S. J. Ram, "Image processing with imagej," *Biophotonics International*, vol. 11, no. 7, pp. 36–42, 2004.

[148] A. C. Green, D. Lath, K. Hudson, B. Walkley, J. M. Down, R. Owen, H. R. Evans, J. Paton-Hough, G. C. Reilly, M. A. Lawson *et al.*, "TGFβ inhibition

stimulates collagen maturation to enhance bone repair and fracture resistance in a murine myeloma model," *Journal of Bone and Mineral Research*, vol. 34, no. 12, pp. 2311–2326, 2019.

[149] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[150] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[151] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. of International Conference on Learning Representations*, 2018.

[152] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[153] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[154] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[155] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[156] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: a comparison to SIFT," *arXiv preprint arXiv:1405.5769*, 2014.

[157] J. Long, N. Zhang, and T. Darrell, "Do convnets learn correspondence?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 1601–1609.

[158] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Proc. of the Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[159] V. Kumar, V. Singh, P. Srijith, and A. Damianou, "Deep Gaussian processes with convolutional kernels," *ArXiv*, vol. abs/1806.01655, 2018.

[160] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[161] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.

[162] D. He, W. Chen, L. Wang, and T.-Y. Liu, "A game-theoretic machine learning approach for revenue maximization in sponsored search," *arXiv preprint arXiv:1406.0728*, 2014.

[163] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[164] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[165] F. Gao, Y. Yang, J. Wang, J. Sun, E. Yang, and H. Zhou, "A deep convolutional generative adversarial networks (dcgans)-based semi-supervised method

for object recognition in synthetic aperture radar (sar) images," *Remote Sensing*, vol. 10, no. 6, p. 846, 2018.

[166] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.

[167] S. Grünewälder, J.-Y. Audibert, M. Opper, and J. Shawe-Taylor, "Regret bounds for Gaussian process bandit problems," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics.* JMLR Workshop and Conference Proceedings, 2010, pp. 273–280.

Appendices

Appendix A

A.1 Uniform Continuity

Given the metric space $(\mathbf{v}, d_1)$ and $(\mathbf{Y}, d_2)$, function f mapping $\mathbf{v}$ to $\mathbf{Y}$ is considered as uniformly continuous, when there is a real slope such that $d_1(\mathbf{v}, \mathbf{y}) < \delta$ for every pairs of $\mathbf{v}$ and $\mathbf{Y}$.

A.2 Modulus of Continuity [1]

In mathematical analysis, a **modulus of continuity** is a function:

$$\omega : [0, \infty] \rightarrow [0, \infty].$$

The uniform continuity of a function is measured by the modulus of continuity. Therefore, a function takes ω as a modulus of continuity if and only if

$$|f(x) - f(y)| \leq \omega(|x - y|),$$

for all x and y in the domain of f.

A.3 Lipschitz Continuity

Given the metric space $(\mathbf{v}, d_1)$ and $(\mathbf{Y}, d_2)$, a function is considered as Lipschitz continuous when a non-positive real constant K exists so that,

$$d_2(f(\mathbf{x})) \leq K d_1(\mathbf{x}).$$

A.4 Dudley's Criterion

Assume that $(\mathbf{v}, d)$ is totally bounded and denote by $N(\epsilon, \mathbf{v})$ the ϵ-packing number of $(\mathbf{v}, d)$, for all positive ϵ. If $\sqrt{\log N(\epsilon, \mathbf{v})}$ is integrable at 0, then $(f(x))_{x \in \mathbf{v}}$ admits a version which is almost surely uniformly continuous on $(\mathbf{v}, d)$. Moreover, if $(f(x))_{x \in \mathbf{v}}$ is almost surely continuous on $(\mathbf{v}, d)$, then

$$\mathbb{E} \sup f(x) \leq 12 \int_0^{\sigma} \sqrt{\log N(\epsilon, \mathbf{v})} d\epsilon, \tag{A.1}$$

where $\sigma = sup \mathbf{Var} f(x)$ is the supremum of the variance on $\mathbf{v}$ [167].

To bound the Dubley integral, we use

$$\int_0^c \sqrt{\log(1 + b\varepsilon^{-\frac{1}{a}})} d\varepsilon \leq c \sqrt{\frac{\log(e2^{a+1})}{a}}, \tag{A.2}$$

which holds for any a, b and c such that $b^a = 2c$. Indeed, letting $\xi = (1 + 2^{-\frac{1}{a}})^a$, we have,

$$\int_0^c \sqrt{\log(1 + b\varepsilon^{-\frac{1}{a}})} d\varepsilon \leq c \sqrt{\frac{\log(e2^{a+1})}{a}}. \tag{A.3}$$

A.5 Equation Derivation

Let $(f(x))_{x \in \mathbf{x}}$ be some centered Gaussian process. Assume that $(\mathbf{x}, d)$ is totally bounded and denote by $N(\epsilon, \mathbf{x})$ the ϵ-packing number of $(\mathbf{x}, d)$, for all positive ϵ. If $\sqrt{\log N(\epsilon, \mathbf{x})}$ is integrable at 0, then $(f(x))_{x \in \mathbf{x}}$ admits a version which is almost surely uniformly continuous on $(\mathbf{x}, d)$. Moreover, if $(f(x))_{x\in\mathbf{x}}$ is almost surely continuous on $(\mathbf{x}, d)$, then

$$\mathbb{E} \sup f(x) \leq 12 \int_0^{\sigma} \sqrt{\log N(\epsilon, \mathbf{x})} d\epsilon, \tag{A.4}$$

where $\sigma = sup\mathbf{Var}f(x)$ is the supremum of the variance on $\mathbf{x}$ [167].

Following Eq.(36)-(43) in [141], we will get

$$\mathbb{E}[\sup f(x)] \leq 12\sqrt{d} \int_0^{\max \sqrt{k(\mathbf{x},\mathbf{x})}} \sqrt{\log(1 + \frac{4rL_k}{\varrho^2})} d\varrho. \tag{A.5}$$

Then following the Dudley integral computations, we can bound the Dudley intergral. We use

$$\int_0^c \sqrt{\log(1 + b\varepsilon^{-\frac{1}{a}})} d\varepsilon \leq c\sqrt{\frac{\log(e2^{a+1})}{a}}, \tag{A.6}$$

which holds for any a,b and c that satisfy $b^a = 2c$.

From Eq. (A.5), we can get that $a = \frac{1}{2}$, $b = 4rL_k$ and $c = \max\sqrt{k(\mathbf{x},\mathbf{x})} = \sqrt{rL_k}$. Then substitute a, b and c into Eq. (A.6), we have

$$\begin{aligned}\int_0^{\sigma} \sqrt{\log(1 + \frac{4rL_k}{\varrho^2})} d\varrho &\leq \sqrt{\frac{\log(e2^{1+\frac{1}{2}})}{\frac{1}{2}}} \max\{\max\sqrt{k(\mathbf{x},\mathbf{x})}, \sqrt{rL_k}\} \\ &= \sqrt{4.0794} \max\{\max\sqrt{k(\mathbf{x},\mathbf{x})}, \sqrt{rL_k}\}.\end{aligned} \tag{A.7}$$