# Short-Term Traffic Participants Behaviour Prediction for Automated Vehicles

by

Mehran Zamani Abnili

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2020

## Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Statement of Contributions

Mehran Zamani was the sole author for Chapters 1, 2, 4 and 5 which were written under the supervision of Prof. Nasser L. Azad and were not written for publication. This thesis consists in part of four manuscripts written for publication. Chapter 3 of this thesis consists of 4 manuscripts co-authored by myself and my supervisor. As lead author of these four manuscript, I was responsible for contributing to conceptualizing study papers, carrying out data collection and analysis, and drafting and submitting manuscripts. My supervisor provided guidance during each step of the research and provided feedback on draft manuscripts.

**Citations:**

M. Zamani Abnili and N. L. Azad, "Short term predictions of preceding vehicle speeds for connected and automated vehicles," Controls, Dynamic Systems and Robotics (CDSR), 2019.

M. Zamani Abnili and N. L. Azad, "A New Data-Driven Approach For On-line Traffic Participant Behaviour Prediction at Intersections for Automated Driving," Progress in Canadian Mechanical Engineering. Volume 3, Jun. 2020, doi: 10.32393/csme.2020.110.

M. Zamani Abnili and N. L. Azad, "Short-term traffic participants behavior prediction using a novel hybrid method in highway merging for automated vehicles," (Under Review)

M. Zamani Abnili and N. L. Azad, "Roundabout Situational Awareness for Automated Vehicles with Hybrid Machine Learning Approach," Controls, Dynamic Systems and Robotics (CDSR), 2020.

**Abstract**

Due to the rapid commencement of autonomous vehicles and the promising potential benefits, it has made it critical for said vehicles to be able to interpret their environment and compensate for the absence of driver predictions from visual cues. This study presents a novel intermediate component to improve the performance of autonomous vehicle controllers, by providing them with real-time microscopic predictions of traffic participants' behaviour, given the environmental conditions. This strategy is especially aimed towards direct combination with model-predictive controllers (MPCs) and other controllers that can utilize dynamic state predictions. This task is undertaken in three stages for three different scenarios.

Scenario I considers V2X communications and predicts the velocity of an arbitrary vehicle in longitudinal direction. Using a recurrent neural network (RNN) and considering a complementary variable the strategy can predict the speed profile of said vehicle for arbitrary horizons. Results of this scenario exhibit $> 0.95$ correlation if trained with enough data.

Scenario II moves on to a more sophisticated approach for prediction of vehicles on US-101, using real data provided by the U.S. Federal Highway Administration (FHWA) under NGSIM. Utilizing a marriage of dynamic Bayesian network (DBN) and RNN, the method can make predictions on speed profiles of all present vehicles within a range, for arbitrary horizons, as well as prediction on whether the vehicle on the main lanes would yield to the merging vehicles on the ramp. Due to digital nature of the DBN stream, a Kalman filter (KF) was introduced as post processing smoothing method. Results of this scenario exhibit $> 0.95$ correlation and $< 1.6$ mph mean absolute error.

Scenario III tackles a much more complex driving environment, intersection driving. Because in intersection driving, the priority relationships of highway driving are no longer existent, the training must be broadened to encompass vehicle pairs which is exponentially more difficult than training for single vehicles. The data for this phase was generated by SUMO. Results of this scenario exhibit $< 1.1$ mph mean absolute error.

Scenario IV focuses on the problem of roundabout driving. In roundabout driving, the general driving situation is more similar to highway merging, however due to the rapid move toward replacing intersections with roundabouts, especially in developing cities, definitely an important scenario to look at. In this scenario SUMO was used for data generation, a new DBN topology was developed and the results yielded exhibit $> 0.89$ correlation.

To evaluate the performance and the accuracy of the proposed method, it was compared with a collection of sequence prediction techniques, including LSTM and GRU. It

was concluded that the DBN-RNN has the best accuracy and performance among these methods.

Validation of the strategy was planned to be done on the scaled autonomous vehicle test platform developed in Smart Hybrid and Electric Vehicles Systems (SHEVS) lab, where driver-in-the-loop hardware was incorporated and the equipment were prepared but due to COVID-19 closures was not realized.

**Acknowledgements**

I would like to express my deepest gratitude to my Advisor, Professor Nasser L. Azad for his patience, dedication, and continuous support throughout my MASc. study and research. His teachings exceed far beyond what is presented in this document and on a personal level has inspired growth, trust, and devotion by being an exemplary supervisor. It was an incredible experience to work with him and I could not have asked for a better advisor.

I am also grateful to my readers Professor Arash Arami and Professor Krzysztof Czarnecki whom their feedback and guidance have proved invaluable to this study. I am truly inspired by their dedication, enthusiasm, and expertise.

Additionally, I would like to thank Professor William W. Melek who introduced me to artificial intelligence and the invaluable opportunity that led to this project and where I am today.

Last but not least, I would like to express my appreciation for my parents, whom without their support, love, and care this would not have been possible, and friends whose company made this journey all the more pleasant.

## Dedication

In the hopes that this work may contribute to safer transportation, I dedicate it to those who have been a victim of traffic accidents.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

**ACC** Adaptive Cruise Controller 3, 4, 41

**ADAS** Advanced Driver-Assistance Systems 4

**ANN** Artificial Neural Network 4, 17

**ARIMA** Autoregressive integrated moving average 4

**BP-Algorithm** Back Propagation Algorithm 53, 85

**CNN** Convolusional Neural Network 6

**CVPM** Continuous Variable Prediction Module 5, 53, 54, 59, 63, 75

**DBN** Dynamic Bayesian Network 4, 5, 13, 16, 17, 25, 33–36, 38, 41, 45–50, 54, 59, 60, 62–64, 75

**EM-Algorithm** Expectation-Maximization Algorithm x, 13, 16, 36, 41

**EPA** U.S. Environmental Protection Agency 3, 4, 25

**eRCNN** error-feedback Recurrent Convolutional Neural Network 4

**GRU** Gated Recurrent Unit 4, 20, 22, 23, 25, 53, 54, 59, 63, 64

**HEV** Hybrid Electric Vehicle 4, 25

**HMM** Hidden Markov Model 4, 13

**k-NN** k-Nearest Neighbors algorithm 4

**LSTM** Long Short-Term Memory network 4, 20–23, 25, 50, 53, 54, 59, 63, 64

**ML** Maximum Likelihood 14

**MPC** Model Predictive Method 4, 25, 36, 41, 50

**NGSIM** Next Generation Simulation 32, 33

**NHTSA** U.S. National Highway Traffic Safety Administration 2

**PGM** Probabilstic Graphical Model 4

**PHEV** Plug-in Hybrid Electric Vehicle 3, 25

**RNN** Recurrent Neural Network 4, 5, 19–21, 23, 25, 35, 36, 38, 45, 47–50, 53, 54, 59, 60, 63, 64, 75, 76

**SAE** Society of Auto-mobile Engineers 2

**SHEVS** Smart Hybrid and Electric Vehicles Systems Lab 5, 67, 68, 73

**SUMO** Simulation of Urban Mobility 46, 61, 62, 72

**SVM** Support Vector Machine 3, 4

**V2I** Vehicle to Infrastructure 4

**V2V** Vehicle to Vehicle 4

**V2X** Vehicle to Anything 3, 24, 34, 47

# Nomenclature

**GIS** A Geographic Information System is a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data. 3, 25

**GPS** The Global Positioning System, originally NAVSTAR GPS, is a satellite-based radionavigation system. 3, 4, 25

**IMU** An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers 68

**LiDAR** LiDAR is a surveying method that measures distance to a target by illuminating the target with laser light and measuring the reflected light with a sensor. 1, 68

**RADAR** RADAR is a detection system that uses radio waves to determine the range, angle, or velocity of objects. 1

**ROS** Robot Operating System is robotics middleware. ROS provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. 70

**VISSIM** PTV Vissim is a microscopic multi-modal traffic flow simulation software package developed by PTV Planung Transport Verkehr AG in Karlsruhe, Germany. The name is derived from "Verkehr In Städten - SIMulationsmodell" 3, 24

# Chapter 1

# Introduction

Human progress and quality of life has seen a massive boost since the dawn of machines, and every day, the rate of this progress is increasing. Machines make tasks easier, safer, more affordable, more precise, and more accessible. There is no doubt that machines have become an integral part of human life, and human experience and we rely heavily on these instruments. One such machine, and arguably the one with the greatest impact on human life, has been the automobile. International Organization of Motor Vehicle Manufacturers estimates that there is about 1.32 billion vehicles in use today and this number is rising every year. Automation, as an augmentation for every machine, has made them even more efficient, faster, and safer. However, automation in driving has been lagging behind due to the immense complexity of the task. It was soon clear that conventional control techniques are not sufficient for a car to be able to navigate the chaotic traffic environment, and soft-computing techniques were not feasible due to inept computing hardware. However, as the computer hardware becomes faster and more powerful, the handicap is being lifted from the implementation of these techniques for this task. Given these achievements in computer domain, and the rise of automation, more and more devices, products and apparatus are introduced to the market, bearing a "smart" or "automated" label. The extra features that make these products deserving of the label, are mostly thanks to a combination of *understanding the environment*, and *being able to act in accordance*.

To understand the environment, autonomous machines are equipped with a battery of sensors and instruments to measure and collect data from their surroundings. These sensors are various in type and are often an arrangement of RGB cameras (conventional cameras), RADARs, LiDARs and sonar range-finders. However, as is often the case, raw measurements from the environment are not sufficient for the robot to navigate, or in general, operate at all. the data must be transformed in a way that is comprehensible and

consumable for a machine. Human drivers have the ability to make decisions in novel situations even if that decision is not the optimal one, normal driving also happens habitually for them. Now considering the case of autonomous vehicles, the reader must acknowledge two facts: Self-driving cars will not be replacing conventional vehicles instantly, therefore, a transition period is imminent and inevitable. As a result, these entities are to adopt driving standards intended for human drivers, legacy of infrastructure layout meant for, and the chaos introduced by humans' inconsistency and erratic behaviour. Another fundamental fact arises from the exact same discord in human behaviour and the fault in aforementioned *sensing* of the environment. Disclosed documents by U.S. National Highway Traffic Safety Administration (NHTSA) [1, 2] for surveys concluded in 2015 indicate that most of the traffic accidents were caused solely due to human error. Out of the 2,189,000 reported accidents, 94% $\pm$ 2.2% were due to human error, and grievously 35,092 were fatal. The same document recounts that a statistically significant portion of aforementioned traffic accidents were due to recognition error (41% $\pm$ 2.2%) followed by decision errors at 33% $\pm$ 3.7% and execution errors at 11% $\pm$ 2.7%. Resultantly, the two principal takeaways would be that:

1. Human drivers struggle at, and fail mostly on recognition (and arguably decision as a consequence) therefore correct recognition is the most difficult element in driving.

2. Whilst in the transition period, the autonomous vehicle must develop judgement that is superior to that of the humans and account for not only itself, but also all other traffic participants in hopes to decrease the accident figures.

To clarify, what is regarded to as an autonomous vehicle in this document, is in fact the fully autonomous vehicle, or one that is at level 5 of autonomy in the Society of Auto-mobile Engineers (SAE) standards (i.e. a vehicle that requires zero human interference in its operation). However, the challenges, especially in the perception domain are prevalent and universal, until every traffic participant is automatized, which is not attainable within the near future. However despite these challenges, researchers do not refrain from investigating this subject, for autonomous driving promises affordable long-range transportation to the public by replacing the human workforce and minimizing energy consumption; Provides a means for personal conveyance for individuals with disabilities, minimizes harmful by-products such as exhaust emissions in case of internal combustion engine-driven vehicles [3] and decrease power plant waste in case of their electric counterparts. Most importantly, autonomous vehicles are expected to minimize traffic accidents and thereupon the property, and tragically in some cases life and limb, associated with such occurrences.

It goes without saying that trying to enumerate all the possible driving scenarios for a machine is quite infeasible. To be able to achieve above-mentioned goals, a blend of

aforementioned soft computing methods need to be used to let the machine develop its own judgement and compensate for the lack of cues only available to human drivers, such as body language, hand signals, high-beam lights and occasional hazard lights for gratitude. However, despite the preceding matters discussed, there seems to be a theme of neglect for the perception for this application in the literature. Commonly, studies in this subject heed to controls as an independent component, or propose solutions that deal with perception and controls in conjunction. The problem with the former is clear and is the main impetus of this study, the issue with the latter is that generally, the approach selected for undertaking such difficult tasks, is also one that is black-box in nature and does not provide any useful detail other than direct control outputs which are not always very reliable. En masse, three general approaches exist and these methods can be classified into three general categories:

1. Classical designs
2. Artificial intelligence and machine learning methods
3. Hybrid approaches

Classical designs usually yield the fastest responses and are the least computationally demanding, and can handle simpler tasks with well-defined models very well. However, there are clear limitations to the classical approaches, namely, their inability to scale well with the complexity of the problem. *Gong et al.* [4] investigated the application of dynamic programming for velocity prediction given knowledge of the environment with GPS and GIS data to minimize fuel consumption. *He* [5] studied linear regression to predict U.S. Environmental Protection Agency (EPA) drive cycles to minimize Plug-in Hybrid Electric Vehicle (PHEV)s power consumption. *Bender et al.* [6] attempted to optimize hybrid hydraulic vehicle's energy management using a rule-based strategy to predict drive cycles. *Chen et al.* [7], designed an Adaptive Cruise Controller (ACC) system that considers movement prediction for other traffic participants by profiling five possible scenarios in which the movement progresses over time.

On the other end of the spectrum are the artificial intelligence, and machine learning methods. These methods are much more versatile and can handle much more complex problems than that of classical designs, they are however considerably slower and more computationally demanding. *Zhang et al.* [8] studied "chaining neural networks" to produce speed predictions, processing VISSIM data, built on Wiedemann's car-following model and assuming Vehicle to Anything (V2X) communications. *Yao et al.* [9], presents a Support Vector Machine (SVM) approach to generate short-term speed prediction for

an "urban corridor," accompanied by a review of multiple prediction methods (SVM, Artificial Neural Network (ANN), k-Nearest Neighbors algorithm (k-NN), Autoregressive integrated moving average (ARIMA)). *Yeon et al.* [10], took an Long Short-Term Memory network (LSTM) approach and compared the results with predictions assuming constant velocity/acceleration. Moreover, *Thorsell* [11] studied Recurrent Neural Network (RNN), LSTM, and Gated Recurrent Unit (GRU) to predict speed profiles. *Wang et al.* [12] took an error-feedback Recurrent Convolutional Neural Network (eRCNN) to predict traffic speed in a macroscopic scale. A similar endeavor was conducted by *Ma et al.* [13], utilizing LSTM to predict the traffic in a Beijing road section using microwave sensor data. In another study, *Moser et al.* [14], studied short-term vehicle velocity prediction using Bayesian networks by assuming Vehicle to Infrastructure (V2I) and/or Vehicle to Vehicle (V2V) communications. In a similar fashion, but in a more macroscopic time domain, *Roos et al.* [15] utilized Dynamic Bayesian Network (DBN)s to predict passenger flow in the rail network of Paris. Furthermore, similar to the intent of part of this study but with simplifying assumptions, considering only single lane highway merging and only yield prediction, *Dong et al.* [16] investigated Probabilstic Graphical Model (PGM) methods. Moreover, *Wang et al.* [17] utilized a Bayesian approach to semantically classify a driver's behaviour based on their physical and psychological thresholds. Last but not least, *Gindele et al.* [18] used a DBN to predict driver behaviour at intersections.

Hybrid approaches aim to fuse multiple methods, such that the weaknesses of one method, be it classical or artificial intelligence are accounted for by the strengths of another. *Sun et al.* [19] compared multiple perception components: exponentially varying, stochastic Markov chain and neural network base, focusing on energy management in Hybrid Electric Vehicle (HEV)s and in company of Model Predictive Method (MPC)s. *Sakhdari et al.*, [20] used least-square parameter estimators for velocity prediction in an ACC developed by a tube-based MPC. MPCs have a lot of potential to combine with prediction components as they have an innate ability to utilize state predictions, and have been studied extensively in the case of autonomous driving or other Advanced Driver-Assistance Systems (ADAS) such as ACC in [20] or [21]. However, in most MPC studies, state predictions are assumed to be given or done naïvely with primitive methods, which creates a big gap in the research. *Murphey et al.* [22] used dynamic programming and neural networks to classify EPA drive cycles based on the driving environment. *Sarkar et al.* [23], used DBN for trajectory prediction of traffic agents at urban intersections, in association with random forests. *Wang et al.* [24] coupled GPS data with social media content and fused it with a Hidden Markov Model (HMM) for a better traffic estimation on a macroscopic extent.

The objective of this study is to find a method that can make valid and accurate predictions for traffic participants motion in different driving scenarios. Of course, for

these predictions to be valid they must meet a certain set of criteria:

- Predictions must be real-time executable
- Predictions must compensate for the lack of human communication
- The method must have the capability to be generalized to encompass different driving scenarios

This study can be classified as a hybrid approach and combines DBN with a Continuous Variable Prediction Module (CVPM), mostly RNN to predict the course, and the behaviour of the ego-vehicle and other traffic participants. It expands on [25], [26] and [27] and focuses on developing a perception strategy for traffic participant behaviour prediction in driving. As mentioned in previous studies, sophisticated prediction strategies can improve the performance of MPCs quite significantly. Most MPC controllers in the literature make simplifying assumptions for the development of the states that are not completely reflective of real-life scenarios. For instance, [20] uses least-square parameter estimators for velocity prediction, [7] considers predefined speed profiles which impair the generalization of the proposed controller, and [28] makes the assumption that the states of all other traffic participants are fully known for the NMPC-based multi-lane adaptive cruise controller. Universally, it is arguable that all of these endeavours would either produce better results or be at all practical, with a decent prediction strategy. Although statistically, most of the traffic accidents are due to recognition and perception errors, it seems that this crucial aspect of automated driving is being neglected in the literature. In this study, a novel data-driven prediction strategy is developed and evaluated for real-time behaviour prediction in a variety of driving scenarios which can also be a complement to MPCs. This strategy fulfils all of the objective criteria and provides means to incorporate context into predictions. In this document, background is discussed in chapter 2, followed by methodology and results discussion, as well as alternative solutions in 3, Smart Hybrid and Electric Vehicles Systems Lab (SHEVS) scaled autonomous vehicle test platform in chapter 4, and finally conclusion and future work in chapter 5.

# Chapter 2

# Background

As discussed in the introduction, approaches to complex problems such as autonomous driving vary, often fundamentally so, and call for breaking down to smaller, simpler chunks that can link together in collaboration and hopefully overcome the challenges and bring forth a breakthrough that can offer fully autonomous vehicles commercially and readily available to everyone. One such reasonable break down is separating perception from controls and having an independent component responsible for making sense of the environment, coming in from sensor measurements and/or raw data from communications. Also discussed in previous chapter is the fact that due to the uncertainties associated with predicting traffic participants' behaviour and the chaotic environment in which they exist, conventional hard-computing methods would most likely fall short. Per contra, machine learning and soft-computing approaches are excellent alternatives that overcome aforementioned deficiencies. Of such methods Bayesian inference and other probabilistic techniques, fuzzy logic, neural networks, genetic algorithm and many others can be listed. While each of these techniques have their unique set of strengths and weaknesses, for driving, one that is able to offer the most amount of knowledge is preferred. As an example, while Convolusional Neural Network (CNN)s can be very powerful tools to find relationships between data, the way they work, does not provide much information about the nature of this relationship or in other words, how the output relates to the input. Contrarily, Bayesian networks, as a product of inferential statistics, can make valid predictions based on a sample of observations, and additionally provide useful information on how the prediction came to be. Another benefit of using Bayesian networks is the probabilistic representation of the predictions. Although the desired output is always the state which has the highest probability, having a the likelihood of an event happening on the side can augment the reliability and the credibility of the inference.

## 2.1 Inferential Statistics

Bayesian networks are receiving increasingly more attention in artificial intelligence research. To understand Bayesian inference and Bayesian networks as their descendent, one must first understand inferential statistics. Inferential statistics is a process in which allows analysing a subset of a large population and make valid predictions by generalization of the sample to the entire population. As an example, a problem of surveying a town of 20,000 populace on whether they use a certain product would be an incredibly difficult and expensive task. However, a way to tackle such problems, one which many enterprises make use of regularly is to survey a smaller sub-population, one of 1,000 to 2,000 people picked at random, and generalize the result to the entire population. It is obvious that the sample size has a reverse relationship with the accuracy of this approach, however a large enough sample size should guarantee an adequate accuracy. Suppose a survey is done over ten days of one hundred people each day to fulfill the task above, the results of which can be found in the table below.

| Day (i) | Number of users ($X_i$) | Number of non-users ($!X_i$) |
|---|---|---|
| 01 | 38 | 62 |
| 02 | 45 | 55 |
| 03 | 41 | 59 |
| 04 | 34 | 66 |
| 05 | 40 | 60 |
| 06 | 49 | 51 |
| 07 | 36 | 64 |
| 08 | 33 | 67 |
| 09 | 38 | 62 |
| 10 | 35 | 65 |

Table 2.1: Proportions of 10 samples of 100 from a population of 20,000

This data grants the ability to calculate the mean and standard deviation, with which any inference can be benchmarked. The mean denoted as $\mu$ can be calculated by summing every sample and dividing the sum by the number of samples denoted as $n$ (2.1)

$$\bar{X} = \frac{\sum X_i}{n} \tag{2.1}$$

Furthermore the standard deviation denoted as $\sigma$ can be computed using (2.2)

$$\sigma = \sqrt{\frac{\sum(X_i - \bar{X})^2}{n}} = \sqrt{\frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + \ldots + (X_n - \bar{X})^2}{n}} \tag{2.2}$$

Standard error of the mean, denoted as $\sigma_{\bar{X}}$ can be computed using (2.3). This equation reaffirms the claim made earlier that the larger the sample size is, the more confidence can be put into the inference and vice versa. The larger the sample size $N$ and the smaller the standard deviation, the smaller the standard error of mean becomes.

$$\sigma_{\bar{X}} = \frac{\sigma}{\sqrt{N}} \tag{2.3}$$

Additionally, the confidence in any hypothesized mean ($\mu$) can be tested using the "z-test" (2.4) and the cumulative standardized normal distribution table in appendix A.

$$z = \frac{\bar{X} - \mu}{\sigma_{\bar{X}}} \tag{2.4}$$

To put above points in perspective, we can apply them to table 2.1. For the sake of argument, suppose after the first day, a rough estimate of 40% users in population was made. After the ten day survey, the mean, standard deviation and other factors can be computed as such:

$$\begin{aligned} \bar{X} &= 38.9 \\ \sigma &= 4.784 \\ \sigma_{\bar{X}} &= 1.513 \\ z &= -0.727 \end{aligned}$$

We will find that 23% of the area of the normal curve will fall below this $z$ value, and since the distribution is two sided, for the offset range it can be computed that another 23% of the area of the normal curve will fall below the $z$ value calculated for $\mu$ of 37.8. In summary, with the expectation of 40% users, and a series of ten samples having a mean of 38.9% the hypothesis can be expected to hold 54% of the time, which does not seem very promising in terms of accuracy. However, salvation comes with updating the initial conjecture with every new observation. This phenomenon was introduced by Rev. Thomas Bayes and was published in [29], two years after his death. What is now known and popular as Bayes theorem is a way to find the probability of an event based on prior knowledge, and will be unfolded in the next section.

## 2.2 Bayes' Theorem and Bayesian Inference

As previously mentioned, classical inferential models do not allow updating hypothesis and introduction of prior knowledge. On the contrary, by using Bayes' rule, the prior knowledge can be included in the computations which can have significant impact on the results. Bayes' theorem can be mathematically formulated as following:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.5}$$

Where $P(B|A)$ is the probability of B given A is true often referred to as "likelihood", $P(A)$ is the marginal probability of A and is often referred to as "prior", and $P(B)$ is the marginal probability of B often referred to as "evidence". The other side of the equation gives $P(A|B)$ and is probability of A given B, often referred to as the "posterior". One example that emphasizes this impact, and its deviation from human intuition is the drug test problem. Suppose a test for using a particular drug is 99% sensitive, and 99% specific. In other words, The test will produce positive result for drug users 99% of the time and negative for non-drug users 99% of the time. Now if the users are very rare, for example 0.5% of the population. What is the probability that a randomly selected individual with a positive test result is actually a drug user? By using (2.5), the problem can be formulated as:

$$\begin{aligned} P(User|+) &= \frac{P(+|user)P(user)}{P(+)} \\ &= \frac{P(+|user)P(user)}{P(+|user)P(user) + P(+|non-user)P(non-user)} \\ &= \frac{0.99 \times 0.005}{0.99 \times 0.005 + 0.01 \times 0.995} \\ &\approx 33.2\% \end{aligned}$$

Which indicates that with a single positive result from a random sample, the chances of the individual being a user is in fact lower than that of not. This is due to the fact that the number of drug users are very small compared to non-users. However, the theorem really shines when the drug test is conducted a second time with a positive result. The hypothesis of the individual being a user has now changed from 0.05% to 33.2%. Formulating the

problem again and updating the hypothesis will give:

$$\begin{aligned} P(User|+) &= \frac{P(+|user)P(user)}{P(+)} \\ &= \frac{P(+|user)P(user)}{P(+|user)P(user) + P(+|non-user)P(non-user)} \\ &= \frac{0.99 \times 0.332}{0.99 \times 0.332 + 0.01 \times 0.668} \\ &\approx 98\% \end{aligned}$$

And subsequently a third positive result will give 99.98% probability that the individual is a user.

Having introduced Bayes' rule, Bayesian inference can be discussed as an extension. Having multiple dependent variables, where the correlation is generally understood, the probability of an event occurring is computable. The computation involves conditional probabilities similar to the left hand side of Bayes' rule, joint probabilities, or a combination of the two depending on the desired outcome. Generally, joint probability can be broken down into conditional and marginal probabilities using (2.6).

$$P(A = a, B = b) = P(a, b) = P(a|b)P(b) = P(b|a)P(a) \tag{2.6}$$

However, often times any real application will consist of more than two variables and the dependencies can be complex. Bayesian networks can resolve and simplify the computations for such scenarios.

## 2.3 Bayesian Networks

Bayesian networks encompass the discussed properties of inferential statistics and Bayes' rule and fuse them with knowledge of the dependencies between the appointed variables. Bayesian networks are represented as acyclic directed graphs in which each node represents a probabilistic variable and each edge serves as the dependency between said variables; as a convention, the child node is directly dependent of the parent node. Bayesian networks have characteristics that can be capitalized on to make the inference more efficient, as exact inference is #P-Hard, and the complexity is $O(2^n)$ for only boolean variables. The advantage of using Bayesian networks to embody the variables can be demonstrated in the example below: Suppose a set of five boolean variables. To find the joint distribution of

| A | B | C | D | E | P(a,b,c,d,e) |
|---|---|---|---|---|---|
| a | b | c | d | e | $p_1$ |
| a | b | c | d | !e[1] | $p_2$ |
| a | b | c | !d | e | $p_3$ |
| a | b | c | !d | !e | $p_4$ |
| a | b | !c | d | e | $p_5$ |
| a | b | !c | d | !e | $p_6$ |
| a | b | !c | !d | e | $p_7$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| !a | !b | !c | !d | !e | $p_{2^5}$ |

Table 2.2: Complete set of settings in joint probability for 5 boolean variables

all variables, a complete permutation of every possible setting must be enumerated, as is demonstrated in table 2.2.

It is clear that to find every joint probability, $2^n$-1 settings must be computed. On the flip side, suppose a Bayesian network in the configuration depicted in Figure 2.1. in this case, the joint probability can be computed as in (2.7)

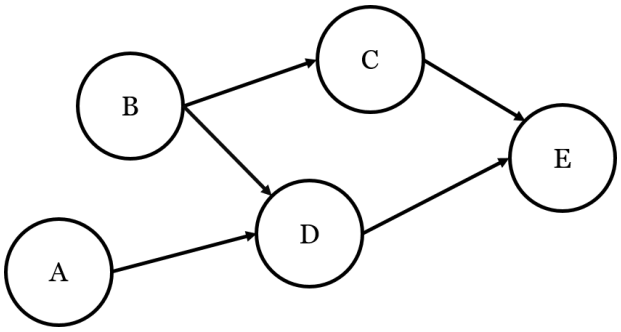



Figure 2.1: Sample Bayesian network topology

$$P(a, b, c, d, e) = P(a)P(b)P(c|b)P(d|b, a)P(e|c, d) \tag{2.7}$$

Which is simplified by accounting for the nature of the dependencies based on the network topology. In this specific topology, only 12 computations are required to find the complete set of joint probabilities. From there to find the marginal probability of any variable taking

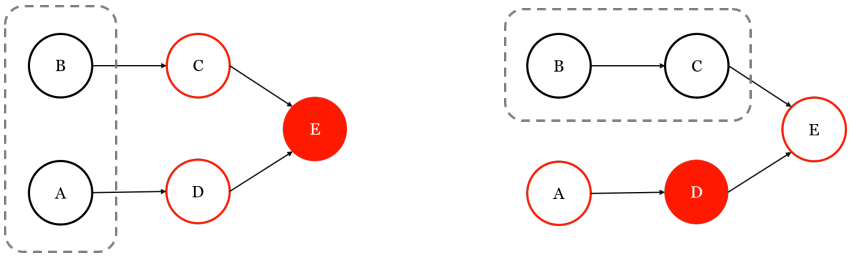
[1] "!" denotes "not"

a specific setting can be computed. For instance the conditional probability of $P(a|e)$ can be computed in (2.8).

$$P(a|e) = \sum_b \sum_c \sum_d \frac{P(a, b, c, d, e)}{P(e)} \tag{2.8}$$

A set of rules that can be capitalized on that will make computation in Bayesian networks more efficient can be found below:

- Each variable is conditionally independent of its non-descendants given its parents. (Figure 2.2)

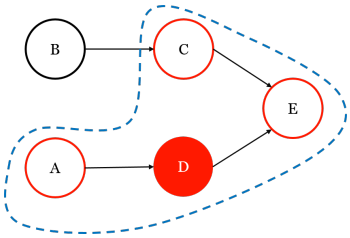



E is independent of A and B given C and D.

D is independent of B and C given A.

Figure 2.2: Conditional independence in Bayesian networks

- Each variable is conditionally independent of all other variables given its Markov blanket. (Figure 2.3)




D is independent of any other node, given A,C and E.

Figure 2.3: Markov blanket independence in Bayesian networks

Bayesian networks can be extended to include nodes that are of a variable in another time-slice. Often, this concept is capitalized on to equip the Bayesian network with ability to predict the state of an existing variable in the next time slice. Such node is in turn termed a "temporal node" and the network is named a "Dynamic Bayesian Network" to be distinguished from a regular Bayesian network; everything else remains the same. Figure 2.4 illustrates a sample topology for the DBN where C' is the variable C in the next time slice.

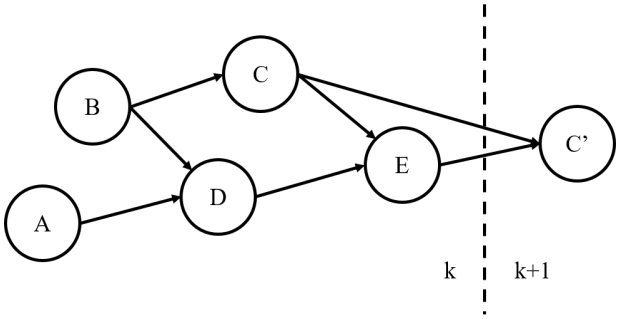



Figure 2.4: Sample dynamic Bayesian network topology

Bayesian networks can be classified as *expert systems* in the sense that the topology is often defined by an expert. Although there are methods that automatically extract a topology from the data, the expert defined topology approach remains prevalent in research. Bayesian networks can also be classified under *supervised learning* solutions in the machine learning communities (i.e. data driven approach). Being a data driven approach, Bayesian networks need to be able to "learn" from datasets. In order to achieve that, an algorithm was developed in the '60s called Baum-Welch algorithm which took advantage of forward-backward algorithm to find unknown parameters of a HMM. Later expanded and developed by Judea Pearl, a more sophisticated algorithm was introduced which was a generalized form of the Baum-Welch algorithm called EM-Algorithm, all being a subsidiary of belief-propagation algorithms, using the Bayes rule. The entire perspective of the EM-Algorithm is to define a set of predetermined size for Gaussian bell curves fitted over a mixture distribution of n-dimension. In other words, for a multi-modal random dataset, the EM-Algorithm will find the parameters of a set number of normal distributions. To understand the inner workings of EM-Algorithm, the following paragraphs is paraphrased from "Learning Dynamic Bayesian Networks" by Zoubin Ghahramani [30].

As mentioned above, the structure of a Bayesian network — set of edges in the topology — and the model parameters are given as *a priori* knowledge. This initial knowledge is considered in the form of a prior probability distribution over model structure and

parameters. The data then updates the prior knowledge to yield posterior probability over models and parameters. Let $P(\mathcal{M})$ be a prior distribution over model structures and $P(\theta|\mathcal{M})$ a prior distribution over parameter for each model structure. Finally, a dataset $\mathcal{D}$ is used to form posterior using Bayes rule. (2.5)

$$P(\mathcal{M}|\mathcal{D}) = \frac{\int P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})d\theta P(\mathcal{M})}{P(\mathcal{D})} \tag{2.9}$$

The integral terms serves to integrate uncertainty over parameters out. For a given model structure, the posterior distribution over parameters can be found using:

$$P(\theta|\mathcal{M},\mathcal{D}) = \frac{P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})} \tag{2.10}$$

If the dataset is a sequence of observations $\mathcal{D} = \{Y_1,\ldots,Y_T\}$ and $Y_{T+1}$ is desired, then the Bayesian prediction

$$P(Y_{T+1}|\mathcal{D}) = \iint P(Y_{T+1}|\theta,\mathcal{M},\mathcal{D})P(\theta|\mathcal{M},\mathcal{D})P(\mathcal{M}|\mathcal{D})d\theta d\mathcal{M} \tag{2.11}$$

integrates out the uncertainty in the model structure and the parameters. If we assume a single model structure $\mathcal{M}$ and estimate the parameters $\hat{\theta}$ that maximize the likelihood $P(\mathcal{D}|\theta,\mathcal{M})$ we obtain a limiting case of Bayesian learning. For a large data set and uninformative prior over the parameters, the posterior $P(\theta|\mathcal{M},\mathcal{D})$ will peak around the maxima of the likelihood, therefore the predictions of a single Maximum Likelihood (ML) model will be similar to those obtained by Bayesian integration over the parameters.

Assuming a data set of independent and identically distributed observations $\mathcal{D} = \{Y_1,\ldots,Y_T\}$, then the likelihood of the data set it:

$$P(\mathcal{D}|\theta,\mathcal{M}) = \prod_{i=1}^{T} P(Y_i|\theta,\mathcal{M})$$

Since we're considering a single model $\mathcal{M}$, we can drop the implicit conditioning from the notation. The ML parameters are obtained by maximizing the likelihood, or equivalently the log likelihood ($\mathcal{L}$):

$$\mathcal{L}(\theta) = \sum_{i=1}^{T} log P(Y_i|\theta)$$

If the observation vector includes all the variables in the Network, then for each term we can find the log-likelihood as:

$$logP(Y_i|\theta) = log \prod_j P(Y_i^j|Y_i^{pa(j)}, \theta^j) \tag{2.12}$$

$$= \sum_j logP(Y_i^j|Y_i^{pa(j)}, \theta^j) \tag{2.13}$$

where $j$ is the index over the nodes of the Bayesian network, $pa(j)$ are is the set of parents of $j$, and $\theta^j$ are the parameters that define the conditional probability of $Y^j$ given its parents. The likelihood therefore decouples into local terms.

In case there are hidden variables, the log likelihood cannot be decomposed as in 2.13. Rather, we find:

$$\mathcal{L}(\theta) = logP(T|\theta) = log \sum_X P(Y, X|\theta) \tag{2.14}$$

where X is the set of hidden variables. $\sum_X$ or $\int_X$ serves to obtain the marginal probability of the data. 2.14 drops the index $i$ and evaluates the log likelihood ($\mathcal{L}$) for a single observation. Assuming a distribution $Q$ over the hidden variables $X$ we can obtain a lower bound on $\mathcal{L}$:

$$\begin{aligned} log \sum_X P(Y, X|\theta) &= log \sum_X Q(X) \ \frac{P(Y, X|\theta)}{Q(X)} \\ &\geq \sum_X Q(X) \ log \ \frac{P(Y, X|\theta)}{Q(X)} \\ &= \sum_X Q(X) \ log \ P(Y, X|\theta) \\ &- \sum_X Q(X) \ log \ Q(X) \\ &= \mathcal{F}(Q, \theta) \end{aligned} \tag{2.15}$$

where the middle inequality is known as Jensen's inequality and can be proven using the concavity of the log function. If the *energy* of a global configuration $(X, Y)$ is defined to be $logP(X, Y|\theta)$ then the lower bound $\mathcal{F}(Q, \theta) \leq \mathcal{L}(\theta)$ is the negative of a quality known as *free energy* or the expected energy under Q minus the entropy of Q. EM algorithm alternates between maximizing $\mathcal{F}$ with respect to $Q$ and $\theta$ respectively, holding the other

fixed.

$$\textbf{E Step:} \qquad Q_{k+1} \leftarrow \underset{Q}{\operatorname{argmax}}\ \mathcal{F}(Q, \theta_k) \tag{2.16}$$

$$\textbf{M Step:} \qquad \theta_{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}}\ \mathcal{F}(Q_{k+1}, \theta) \tag{2.17}$$

To simplify everything from the beginning, suppose a simple 1-dimensional mixture model with two modes: as illustrated in Figure 2.5, the process can be started by assum-

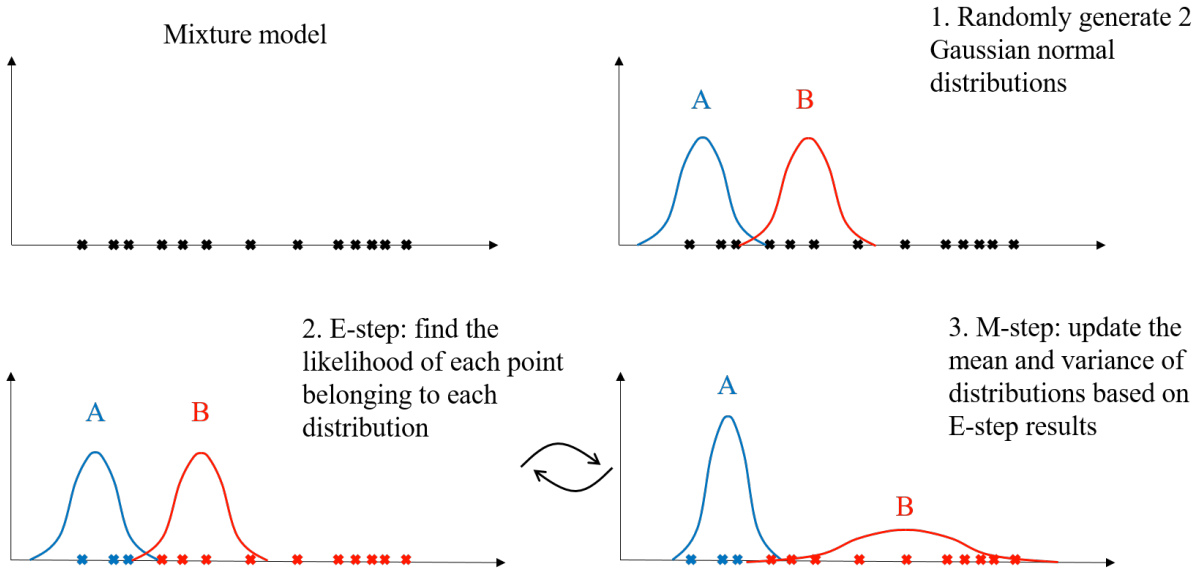



Figure 2.5: Simple 1-dimensional case of EM-Algorithm

ing two randomly generated Gaussian normal distributions. In the E-step, the likelihood of each point belonging to each distribution is computed and each point is assigned to one of the two distributions. In the M-step, the parameters of the Normal distributions are updated to match that of the points. By iterating between E-step and the M-step, distributions A and B will fit the data.

And finally, to use the Bayesian network, or it's temporal counterpart, DBN, the conditional probability of every setting for a variable given its parents are obtained after the training is complete using the EM-Algorithm. Additionally, joint probabilities can be found using 2.6, and finally, marginal probabilities can be found by swapping the denominator with the left hand side of the 2.8.

Using the DBN has a few key benefits enumerated below:

- Probabilistic representation gives better understanding of the situation.
- It provides versatility and the freedom to tailor the topology to a specific application.
- It is intuitive and close to human inference.
- It provides excellent noise rejection as a result of probabilistic representation.

however, there is a particular downside to using DBN, that is its tendency to digitize variables into a handful of classes. This is not a problem when dealing with multi-modal mixture distribution or continuous variables where a semantic classification is sufficient (e.g. {near, far} for distance). However, when dealing with variables such as velocity where the exact quantity is desired, DBNs fall short. Throughout this document, the former is referred to as "high-level variables" and the latter as "low-level variables". Figure 2.6 illustrates a schema of ideal low level and high level variables.

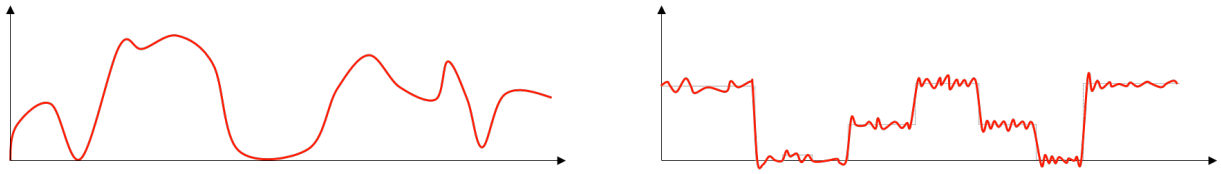


Figure 2.6: Ideal high level variable (right) and low level variable (left)

To overcome the shortcomings of the DBN in regards to low level variables, the task of prediction can be transferred to another data driven machine learning method such as neural networks for said variables. There are of course a multitude of methods that can undertake such task, but recurrent neural network are the go-to approach for time-series.

## 2.4 Recurrent Neural Networks

To understand the mechanism of recurrent neural networks, one must first comprehend the workings of an artificial neural network, a perceptron. Artificial neural networks attempt to clone the physical structure and functionality of their biological counterparts. ANNs consist of the fundamental parts of a biological network, namely, synaptic and somatic operations. Synaptic operation assigns a weighted significance to the inputs according to the knowledge already stored in the synapses, and somatic operation provides a non-linear activation to the neuron. Again, similar to their biological counterparts, ANNs receive inputs from outside or through other neurons in the network and then generate a

product term which will be discussed in the following paragraphs. Neural networks can be characterized in terms of:

- Network size: essentially number of neurons and their layout in the network.
- Neuron activation functions: the non-linear function in each neuron.
- Network pattern: the layout in which the neurons are connected.
- Learning algorithm: how the knowledge is stored in the network.

Given the activation function $\varphi(.)$, each neurons sums the weighted inputs to it and passes it through as the input argument to $\varphi(.)$. In other words:

$$y(t) = \varphi(\sum_{i=0}^{r} w_i x_i - w_0) \tag{2.18}$$

where $i$ is a counter for input connections, $r$ is the number of connections from the previous layer to the neuron and $t$ is the neuron identifier. Figure 2.7 illustrates the workings of a neuron and (2.18).

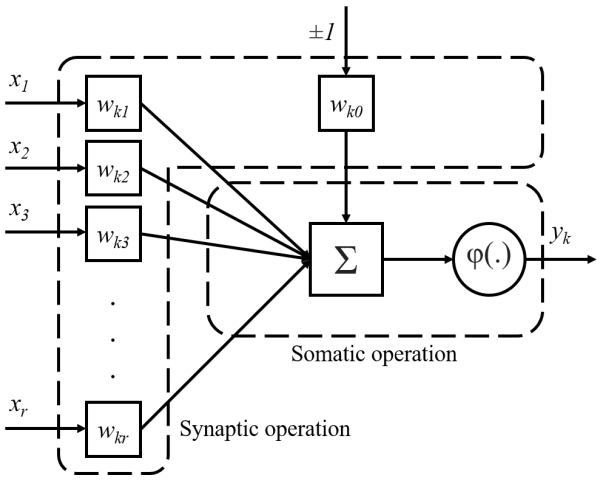



Figure 2.7: Schematics of a neuron

These neurons can then connect together and form a network, where the inputs to each neuron is either the input to the network or the output from previous layers, as illustrated in Figure 2.8. As for the activation function ($\varphi(.)$) there are three main types of functions, Gaussian, linear and sigmoid. In this study, the network structure was chosen to be a

recurrent neural network, for activation functions logistic sigmoid function was selected, and the training procedure was chosen to be error back-propagation, therefore the focus on the coming paragraphs will be on the mentioned.

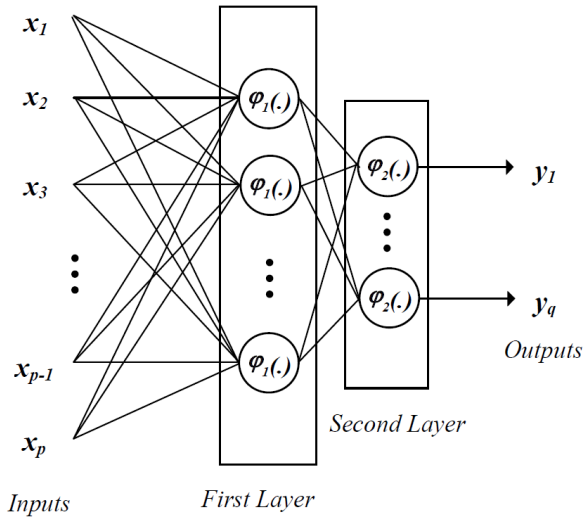



Figure 2.8: Schematics of a neural network

### 2.4.1 Recurrent neural network structure

To incorporate time series, required to make predictions for a dynamic variable, given knowledge of a short history window, the neural network can be equipped with feedback loops. This new structure chains an arbitrary but specific number of inputs to represent a vector and extracts the pattern from them. Figure 2.9 illustrates the schematics of a RNN with two dynamic variables. The network is trained the same way as discussed in Appendix B, however for every generated output, the history window moves one step forward in time. This operation can be repeated indefinitely to predict any arbitrary horizon, however it is of no surprise that every time this operation is repeated, the accuracy drops and after a certain point the accuracy will fall below a desirable threshold. The history window ($d$), the sampling rate, and the network size are the free parameters that can be adjusted to enhance the precision of the approach.

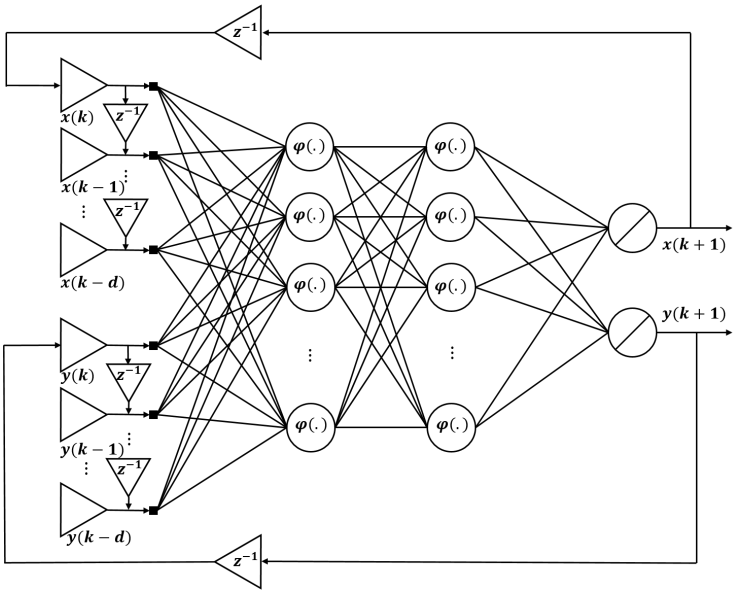



Figure 2.9: Schematics of a recurrent neural network for two dynamic variables

### 2.4.2 Long short-term memory networks

With vanilla RNNs, a common issue is the vanishing or exploding gradient problem. The problem put simply looks at the sequence of weight multiplications and concerns with the recursion of larger than 1 numbers growing unbound and smaller than 1 numbers resulting in 0. This is a problem as the weight updates are proportional to the sum of weights multiplied by the errors of layers closer to the output. As the errors of these layers shrink during training, the weight updates for layers closer to the input may come to a halt and stop the network from training altogether. For networks using ReLu activation function and other activation functions where their outputs can be larger than 1, the opposite may be the case, where the weight updates become divergent. This problem was the motivation behind LSTM and GRU.

With vanishing/exploding gradient problem at hand, evolved RNNs were introduced. One of these networks is the long short-term memory network. These networks, on top of learning to predict a sequence, will learn which parts of a sequence are more important and is able to weigh the stream based on its importance, keep what is useful and *forget* what is not. This is done by introducing three gates into the structure which us a sigmoid activation function for their state to control the flow of the data. The data itself is normalized by tanh activation functions.

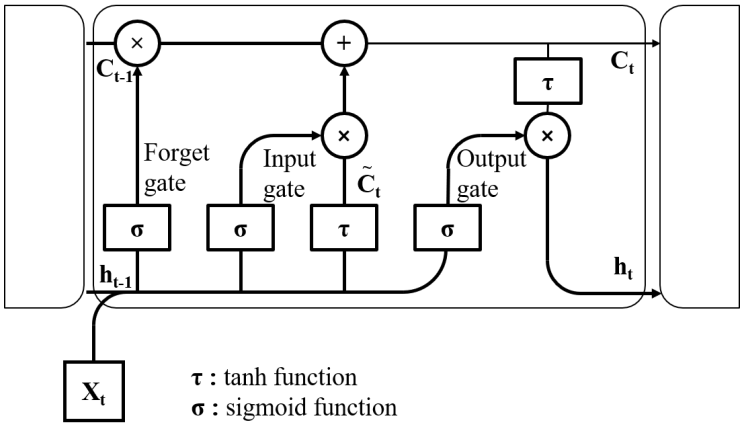



Figure 2.10: Schematics of an LSTM cell

In Figure 2.10, the structure of an LSTM is illustrated. It is clear immediately that here there are two flow paths as opposed to the one in RNN. In LSTMs one path is used for data flow and the other for cell state internal information. The cell state information configures neighbouring cells as of what to consider important and what to let in or out. The most common configuration of LSTMs have 3 gates: forget, input, and output.

1. Forget gate: the leftmost sigmoid function in the schematics is called the forget gate and its function is to decide whether information at time t is worth storing or it should be gotten rid of. The inputs of this gate are the measurements $X_t$ and the hidden states from the previous time step t-1, $h_{t-1}$. These two values will determine an output for the sigmoid function between 0 and 1. Since the output is a element-wise multiplication at the output, an output of 0 will result in 0 cell state at this checkpoint. This will completely ignore the memory going forward.

2. Input gate: the next gate in the chain is called the input gate and is the sigmoid function on the right of forget gate. This gate will decide what data from the combination of $[X_t, h_{t-1}]$ is added to the cell state.

3. Output gate: finally, the rightmost sigmoid function will control the output flow of the information to give the new hidden state.

Mathematically LSTM can be represented by the following equations:

$$\textbf{Forget gate: } f_t = \sigma(W_f.[h_t, X_t] + b_f)$$
$$\textbf{Input gate: } i_t = \sigma(W_i.[h_{t-1}, X_t] + bi)$$
$$\textbf{New candidate values: } \tilde{C}_t = \tau(W_C.[h_{t-1}, X_t] + b_C)$$
$$\textbf{New cell state value: } C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$\textbf{Output gate: } o_t = \sigma(W_o[h_{t-1}, X_t] + b_o)$$
$$\textbf{New hidden state: } h_t = o_t * \tau(C_t)$$

### 2.4.3 Gated Recurrent Unit

Similarly, gated recurrent networks are evolved recurrent neural networks with gates that work much like that of the LSTM, however, two main differences between the LSTM and the GRU is that unlike LSTM, the cell state does not have a separate flow in GRU, also the gates in GRU are one less than that of LSTM but with much of the same functionality. In GRU the gates are named update, and reset. Figure 2.11 illustrates the schematics of a GRU cell.

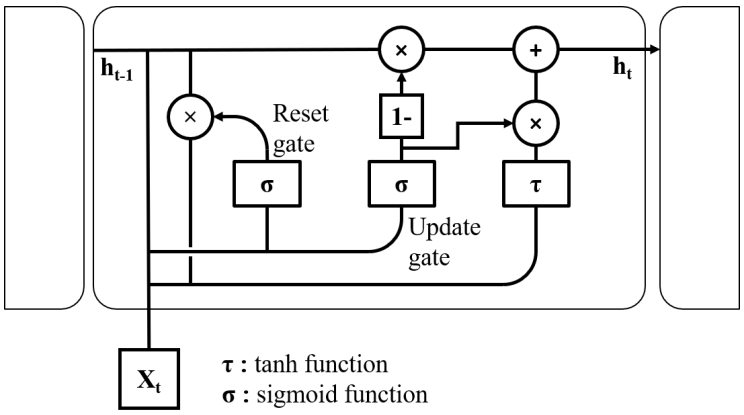



Figure 2.11: Schematics of an GRU cell

1. Reset gate: similar to "forget gate" in LSTM, the reset gate controls what information should be saved to memory and what information should be gotten rid of.

2. Update gate: update gate controls the flow of information after the tanh function and in the unit itself simultaneously. The functionality is similar to that of "input" and "output" gates in the LSTM but with different techniques.

Training in both GRU and LSTM are done using stochastic gradient descent techniques such as SGD, RMSProp, or ADAM.

## 2.5 Kalman Filtering and Kalman Smoothing

In this study, a Kalman filter with a dummy model was used to smooth out the jagged behaviour of the RNN output. The design of the Kalman filter will be discussed in depth in the coming chapters but the mathematical equations can be found below.

For the linear discrete state-space model:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + \epsilon \qquad (\epsilon \sim N[0, R]) \\ y_k &= Cx_k + Du_k + \delta \qquad (\delta \sim N[0, Q]) \end{aligned} \tag{2.19}$$

$$\begin{aligned} \textbf{Prediction Step: } \hat{x}_{k|k-1} &= A\hat{x}_{k-1|k-1} + Bu_{k-1} \\ P_{k|k-1} &= AP_{k|k}A^T + R \end{aligned} \tag{2.20}$$

$$\begin{aligned} \textbf{Correction Step: } K_k &= P_{k|k-1}C^T(CP_{k|k-1}C^T + Q) \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \\ P_{k|k} &= (I - K_kC)P_{k|k-1} \end{aligned} \tag{2.21}$$

# Chapter 3

# Methodology and Results

In this chapter, the progress of work, the methodology, and the results for each phase of the project is discussed. The studied scenarios in this thesis are motion prediction in lack of context considering V2X communications using EPA drive cycles, context incorporation for highway merging using real data, a complex driving scenario, intersection driving using simulated data, and roundabout driving. Although this method can be extended to process any variable, the variable focused here is speed prediction. Incorporating context into predictions is the gateway to gaining the ability to predict a path for each traffic participant. Given a path, start position and speed predictions, each participant can theoretically be tracked and its microscopic motion predicted.

## 3.1 Scenario I: City Drive Cycles [1]

In the first scenario the problem of velocity prediction for an arbitrary traffic participant given knowledge of the environment was studied. Given the interest for having accurate predictions for surrounding traffic participants behaviour, particularly vehicle speeds in this case of autonomous driving, several studies have been conducted in this area. To remind the reader of the studies similar to this scenario, some of the references are restated. Zhang et al. [8] studied chaining neural networks (CNN) to process VISSIM data, built on Wiedemanns car following model, by assuming availability of V2X communications.

[1] This is an accepted manuscript published by Avestia in the Proceedings of the 6th International Conference on Control, Dynamic Systems, and Robotics (CDSR'19) on June 6th 2019, available online: http://doi.org/10.11159/cdsr19.132, M.Zamani Abnili, N.L. Azad, "Short Term Predictions of Preceding Vehicle Speeds for Connected and Automated Vehicles"

A broader study was conducted by Sun et al. [19] to compare three different velocity prediction strategies: exponentially varying, stochastic Markov chain and neural network based. Focusing on the energy management of HEVs, and also the application of the mentioned predictors for MPCs. Gong et al. [4] took another approach and used dynamic programming method to predict the velocity by utilizing knowledge of the environment a priori using GPS and GIS data to minimize fuel consumption. He [5] took a linear regression approach to predict the EPA drive cycles to minimize plug-in HEVs (PHEVs) power consumption. Murphey et al. [22] utilized neural networks and dynamic programming to classify the driving environment using various standard drive cycles from EPA. Bender et al. [6] took a rule based strategy to predict drive cycles for hybrid hydraulic vehicles and also aimed to optimize the vehicles energy management with the results. Finally, Thorsell [11] took a neural network approach to predict speed profiles, specifically RNNs, LSTMs and GRUs. In this scenario a RNN is utilized to make predictions for various horizons on the speed of a preceding vehicle.

### 3.1.1 Data

The focus of this scenario is mostly to find a strategy to predict the movement of surrounding vehicles and does not concern hardware choices. The assumptions made for the data stream from sensors, is that the ego-vehicle is a CAV adequately equipped, to be able to measure the velocity of its peers, and is also given an estimate of its distance from traffic signals that may force a full stop upon it. It is understood that at least one companion variable with correlation to the variable of interest can increase the performance of this method significantly, hence the choice of 'distance-to-stop-sign' alongside the velocity variable. Considering the previous chapters, it is obvious that the DBN is missing in this scenario. By replacing the distance-to-stop-sign with the outputs of a DBN, the prediction method would be more sophisticated and can consider many more case scenarios. This marriage is discussed in the next scenarios.

The datasets used for training of the method are acquired from the EPA. Illustrated in Figure 3.1, the datasets are titled: EPA Urban Dynamometer Driving Schedule (often referred to as UDDS, LA-4 or the city test), New York City Cycle Driving Schedule, and finally, LA92 Unified Dynamometer Driving Schedule. These datasets are especially well-suited for this application, as they maintain the stop and go situation throughout, while introducing variation in the overall behaviour and tone of driving. These variations come in both length of action (e.g. time spent stationary), and intensity of action (e.g. magnitude of velocity). Some pre-processing had to be done on the datasets to prepare them to be used in the neural network training process. The preprocessing was up-sampling the

dataset to 10Hz (10 times more than the original) using polynomial interpolation.

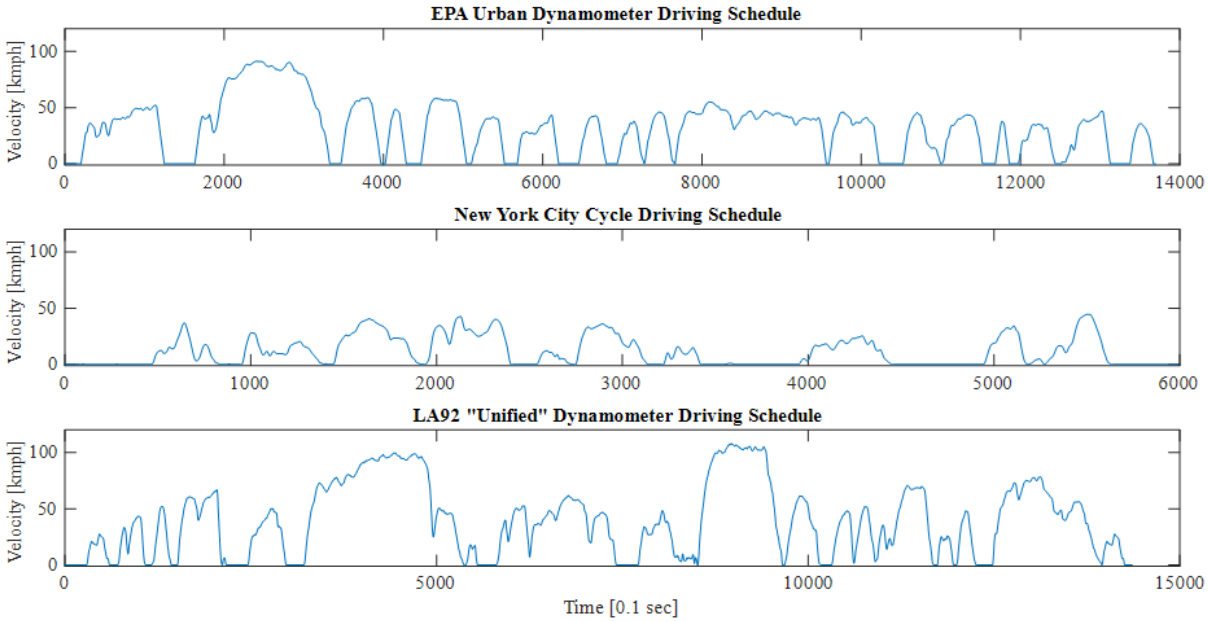
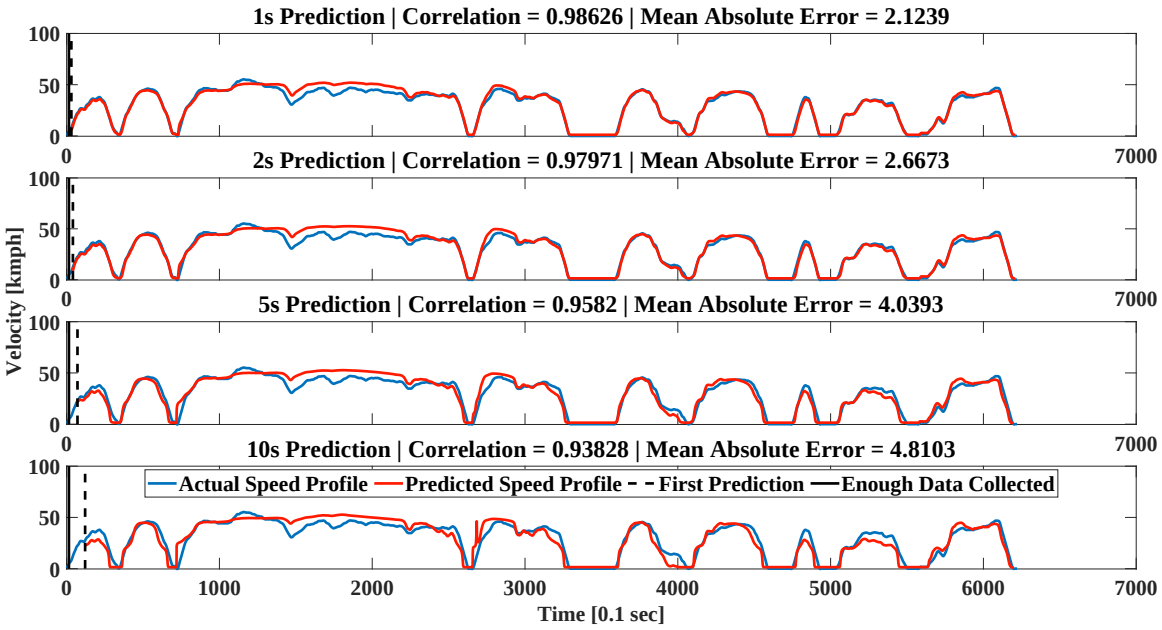


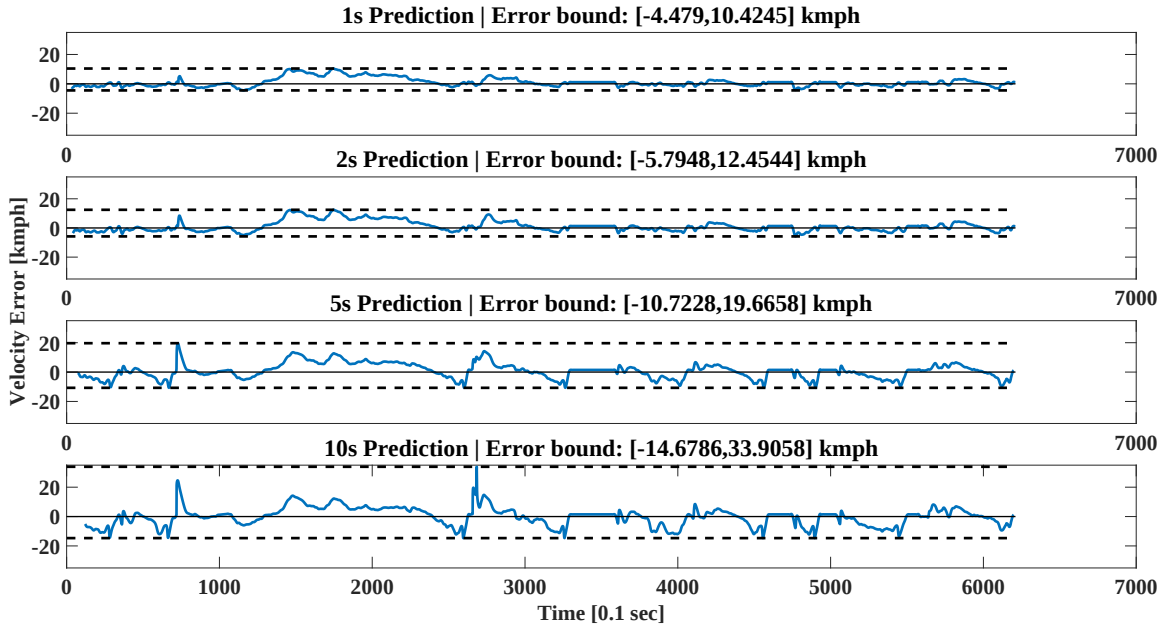

Figure 3.1: Datasets for Scenario I

### 3.1.2 Results

The network predictions have been illustrated in Fig. 3 to 5 for 1, 2, 5, and 10 second horizons. The network trained with the UDDS data seems to capture the changes in the speeds relatively well (Figure 3.2a), especially for smaller horizons. From 700 deciseconds to 2500 deciseconds, the network prediction is overestimating the speed of the vehicle for the most part. That is due to the fact that after integrating the velocity to find the stop sign locations, the area underneath was very large, therefore for the network the next stop sign is too far for the vehicle to decrease its speed. This can be alleviated by replacing the accompanying variable with a different value, or capping its value to a threshold. Although capping the distance-to-stop-sign may increase the accuracy of velocity predictions, finding the optimal maximum value would be a challenge, also this could result in loss of useful information that would have otherwise been provided to the network. The general trend is that as the prediction horizon increases, the accuracy decreases, which is expected. The metric used to assess the accuracy of the network was the correlation between the actual data and the prediction, as well as the mean absolute error. The error bounds are illustrated in Figure 3.2 through Figure 3.4




(a) Network results



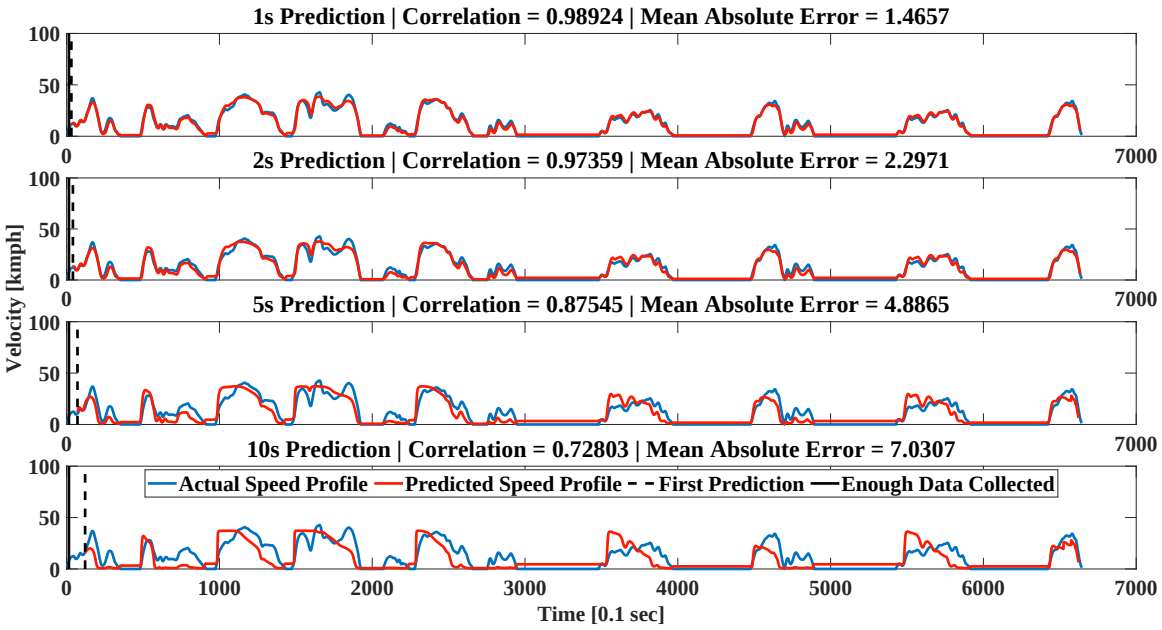

(b) Errors

Figure 3.2: EPA Urban Dynamometer Driving Schedule

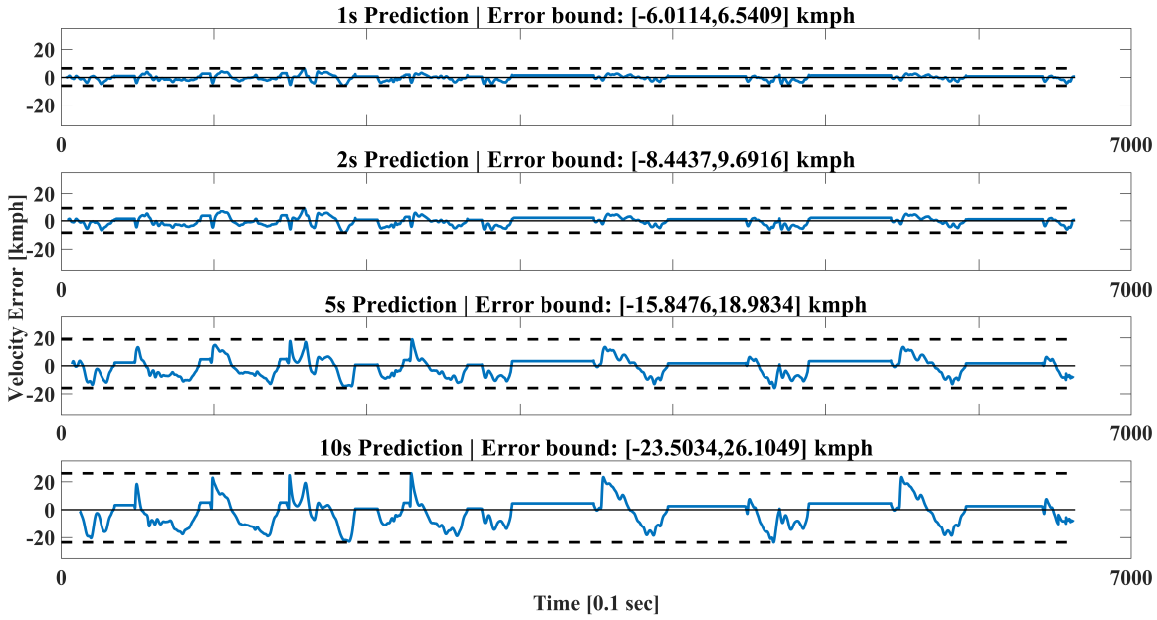
In the New York City Cycle Driving Schedule (Figure 3.3), more loss in the accuracy is observed as the prediction horizon increases, which is due to having less correlation between the main variable, velocity, and the accompanying variable, distance-to-stop-sign. In other words, the accompanying variable does not capture the variations in the velocity due to integration. However, for shorter prediction horizons, the accuracy is still relatively high.

The Results for LA92 (Figure 3.4) are very similar to that of UDDS but with higher accuracy due to larger data. The over-estimation of velocity in the predictor in sections with large area under the curve is again apparent due to the same reason.

Considering the error figures and comparing them to the prediction results, it is apparent that errors peak at instants in which there is a sudden change in the velocity, but then soon after the change is captured by the network, errors diminish. This behaviour is evident in the results for New York City drive cycle, due to lower number of sample points. In general, neural networks tend to rely heavily on the quality of the data that they are trained with. In this application, the best kind of dataset is one that is large, and consists of a fair amount of variation in both intensity, and duration of action. This is supported by superior results yielded by LA92 and UDDS datasets. The significance of this is especially pronounced when compared to the approach taken in [8]. For a one-dimensional car-following application, variations in magnitude of speed is either completely removed from the process or is minimally introduced in a variable fixed-maximum velocity manner (i.e. when the vehicle speed reaches its local maxima, it will continue with the same speed until it is forced to stop), which although relatively simplifies the problem, in turn cripples the method in terms of generalization. Considering the absolute values for velocity and general motion, grants the benefit to take advantage of this method in considerably more cases and scenarios, such as merging in highways, intersections, and roundabouts. Another substantial point to consider is loss of accuracy as prediction horizon increases. In like manner, this can be mitigated by increasing the size of the training dataset. As the training data grows, larger horizons gain more validity. The accuracy loss due to increased prediction horizon is also evident in the results of [19] as the predicted velocity is consistently divergent. Although this may be due to different plotting styles, a pairwise comparison between the results of the method discussed in this scenario with results of the neural network based method in [19] show a fundamental difference. In [19], the results exhibit a kind of passiveness as the predictions keep advancing in the same course. Whereas in this scenario, apparent in the peak of error in UDDS results at 2600 decisecond of 10 second prediction horizon, it seems as if the network is actively correcting its results. Additionally, although training times may be trivial for the aforementioned datasets as the training is done offline, they are a good indication of scalability of the solution and can be

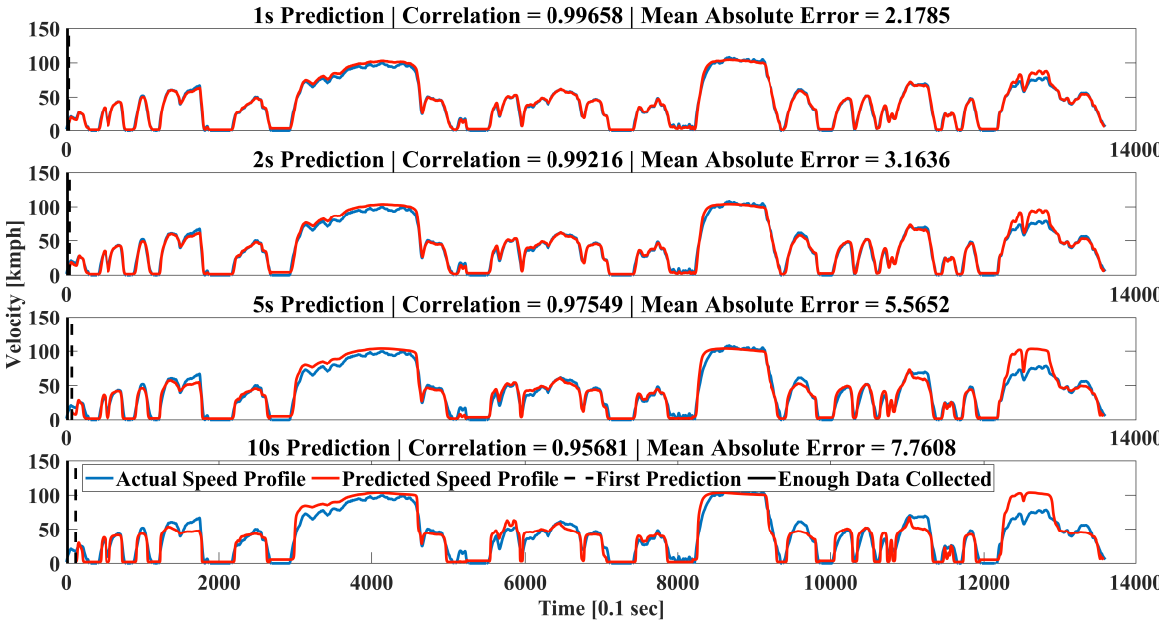



(a) Network results

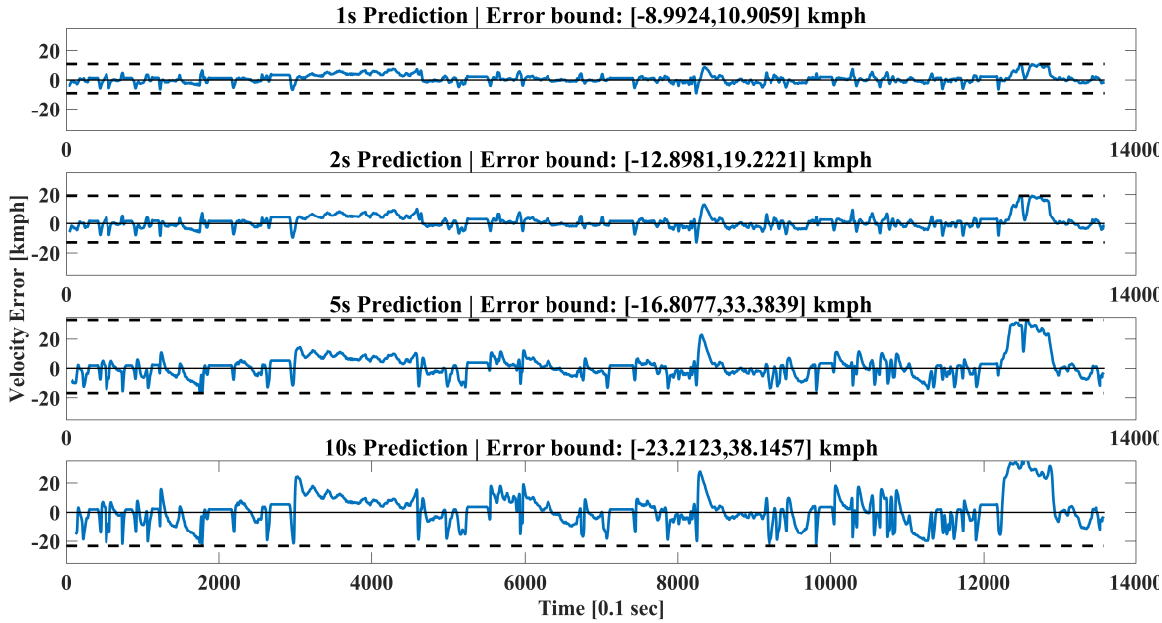



(b) Errors

Figure 3.3: New York City Cycle Driving Schedule




(a) Network results




(b) Errors

Figure 3.4: LA92 Unified Dynamometer Driving Schedule

found in Table 3.1.

| **Dataset** | UDDS | NYCC | LA92 |
|---|---|---|---|
| **Time[s]** | 33972.6 | 25022.4 | 42761.3 |
| **CPU** | Intel® Core ™ i7 5820K @ 3.4GHz | | |

Table 3.1: Training times

Moreover, the turnaround time for 10 second prediction horizon was on average ≈0.0246 seconds (40.65 Hz), with very low deviation, which with 10Hz sampling rate makes it real-time applicable with a large headroom. The results of this scenario was presented at Conference of Controls, Dynamic Systems, and Robotics, Ottawa [25].

## 3.2 Scenario II: Highway Merging [2]

Supervised learning methods, especially neural network type approaches are able to identify patterns in the dataset and learn to replicate them for fresh inputs. Driving rules guarantee the existence of such patterns with limited discrepancy. However, neural network type approaches act in a black-box manner and do not provide much control over the inference nor do they supply any useful information on how the output is generated. Ideally, neural networks are used for MISO (multi-input, single output) functions to generate an output based on the inputs, where the I/O relation is generally understood. Dynamic Bayesian networks on the other hand, although being of supervised learning breed, are incredibly versatile in terms of control between I/O relation. As mentioned previously, for hybrid approaches, the goal is to mix and match different methods such that the weaknesses of one are compensated for by the strengths of another. This scenario capitalizes on the fact that DBNs and neural networks complement each other in a way that creates a viable and potent hybrid approach for the task. Recurrent neural networks with history inputs were explicitly used for their 'memory' and time-series operation ability. Forthcoming sections demonstrate the qualities of this marriage in more detail.

[2]This is an accepted manuscript published by University of Prince Edward Island. Robertson Library in the Proceedings of the Canadian Society for Mechanical Engineering International Congress (2020) on June 24th 2020, available online: http://doi.org/10.32393/csme.2020.110, M.Zamani Abnili, N.L. Azad, "A New Data-Driven Approach For On-line Traffic Participant Behaviour Prediction at Intersections for Automated Driving"

### 3.2.1 Methodology

Highway merging is one of many situations in driving that requires a lot of attention, as errors can be disastrous. Human drivers that are travelling on the highway may react differently in the proximity of an on-ramp with a queue of vehicles intending to merge with the traffic. These discrepancies in behaviour, combined with the fast-paced environment, makes it very difficult for controllers to take safe actions spontaneously. Hence the focus of this scenario is to predict the behaviour of vehicles driving on the highway, for a supposed automated controller on the merging vehicle, to replicate safe scenarios conducted by human drivers. For this purpose, real traffic data was required with the guarantee that none of the actions lead to a collision, therefore the data collected for the Next Generation Simulation (NGSIM) project was used [31].

#### 3.2.1.1 NGSIM Data Preparation

NGSIM data [31] contains traffic data for 640m of US101 highway (south-bound) at Ventura Blvd. ramp and is commonly used by researchers working on highway merging problem. The dataset is very large ($25 \times 11,850,527$) containing information for 3400 vehicles. At the same time the dataset is very messy with missing data. To clean up the dataset, a sliding window search was carried over the entire dataset extracting the data for 100 vehicles at a time and sorting the samples in chronological order. Duplicate IDs were then identified and parsed based on location. Because for every vehicle selected as "ego-vehicle", the information of all other traffic participants must be known synced in global time, every vehicle's information was cut to fit the hypothetical ego-vehicle's presence window and a cell matrix of 34003399 tables. This cell matrix contains all the possible vehicle pairs. To ensure equal size, every offset time was filled with NaNs. After this stage, useful information could be extracted from the dataset which is discussed below. The structure of prediction network was designed to encompass the useful information within the data available. For the DBN, 5 main variables were extracted, two of which are temporal nodes in the network. The set of variables considered are defined below.

**Approaching Flag ($App^P$)** The *Approaching Flag* variable looks at how the distance between the ego-vehicle and other traffic participants is changing over time. If the distance is decreasing, $App^P$ is 1, and 0 otherwise.

$$App^P = \begin{cases} 1 & \frac{\Delta D_{rel}}{\Delta t} < 0 \\ 0 & \text{otherwise} \end{cases}$$

**Acceleration Flag ($Acc^P$)** In contrast to $App^P$, *Acceleration Flag* considers the absolute motion of other traffic participants and outputs 1 for positive acceleration and 0 otherwise.

$$Acc^P = \begin{cases} 1 & \frac{\Delta v_{abs}}{\Delta t} > 0 \\ 0 & \text{otherwise} \end{cases}$$

**Lane Identity ($Lane_{ID}$)** NGSIM data contains traffic data for five lanes plus an auxiliary lane. The section starts close to the on-ramp from Ventura Blvd. and ends close to the off-ramp. One assumption made in the data preparation process was that the 3 lanes farthest from the on-ramp have a negligible impact on the merging vehicle, whereas the first two lanes are the most important. The effect of the first adjacent lane to the auxiliary lane is clear and is extensively studied. The effect of the second lane is also significant, as some vehicles tend to perform a lane change as they notice a vehicle on the on-ramp, to give way. Moreover, it is also beneficial to have smaller sets for the values of each variable as inference in DBN scales exponentially. Therefore, the set of values for $Lane_{ID}$ is defined as:

$$Lane_{ID} \in \{1, 2, Aux, R_{on}, R_{off}\}$$

**Zone Identity ($Zone_{ID}$)** $Zone_{ID}$ is a complement to $Lane_{ID}$ variable. Where $Lane_{ID}$ looks at the lateral position of each vehicle, $Zone_{ID}$ observes their longitudinal position. Although the section arrangement is arbitrary, in this study, the area is divided into four main sections. The area labelled **2** is the area on the first main lane starting from where the ego-vehicle intends to merge to the traffic from the auxiliary lane, to where it successfully merges with the traffic. Area labelled **1** is the area on the first adjacent lane before **2** for an arbitrary limited range, and similarly, **3** is the area after **2**. Area labelled **4** is the area outside **1, 2 and 3**. Figure 3.5 illustrates a schematic of this variable.

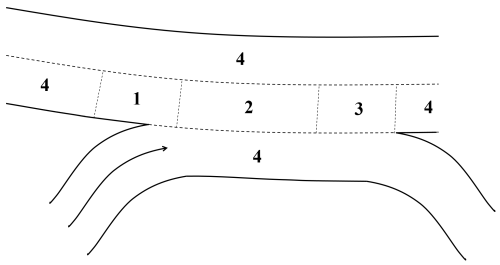



Figure 3.5: $Zone_{ID}$ variable schematic

**Yield Flag ($Yield^P$)** Unlike other variables that were immediately measurable from the environment, $Yield^P$ is a desired output from the DBN and is extracted from the dataset for supervised learning. To extract this variable from the dataset, all the scenarios in which yield is not applicable were set aside, then for the scenarios in which yield is germane, data is labelled. Therefore, the set of values for this variable can be defined as:

$$Yield^P \in \{0, 1, N/A\}$$

#### 3.2.1.2 Dynamic Bayesian Network Topology

The network topology relies deeply on the training data available, as well as the sensor composition on the ego-vehicle (in case V2X communication is available, the set can be expanded to incorporate that data as well). The objective is to transform raw measurements into useful information by exploiting the correlation between variables. Given the designed variables discussed above, and the main intention being traffic forecast, the designed network can extract network inputs from easy-to-measure data. $App^P$, $Acc^P$, $Lane_{ID}$ *and* $Zone$, are source nodes in the directed graph (i.e. nodes with in-degree of 0). $Lane_{ID}$ and $Zone$ are defined as temporal nodes to predict lane and zone changes. Ultimately, yield is defined as a sink (i.e. node with out-degree of 0) in the DBN topology as the final product. The set of edges define the dependencies between variables. As an example, lane change often comes coupled with changes in acceleration (or lack thereof) and depends on the prior lane the vehicle is travelling on. Figure 3.6 illustrates the designed topology of the DBN for this application.

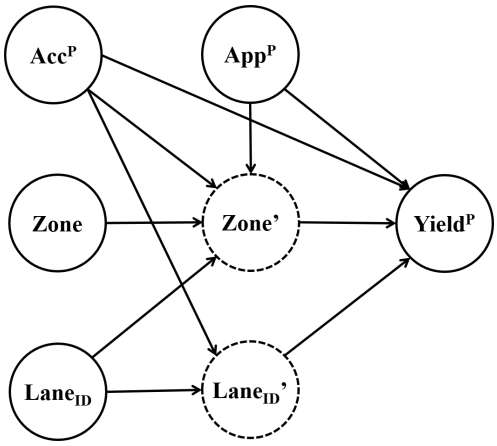



Figure 3.6: DBN topology

#### 3.2.1.3 DBN-RNN Combination

As mentioned previously, the performance of an RNN in traffic prediction relies heavily on the complementary information that is provided to it from the environment. The point of this extra information is to grant the network with some cues of a variable's state in the future. Also mentioned previously is the fact that DBNs are not as effective as other methods when it comes to dealing with continuous variables, the likes of which are frequent in traffic predictions. In this research, velocity was the variable with such qualities. A recurrent neural network can substitute *EM algorithm* inference in any of the DBN vertices to alleviate the shortcoming associated with continuous variables. Figure 3.7 illustrates a schematic of the connection between DBN and RNN with the RNN node expanded.

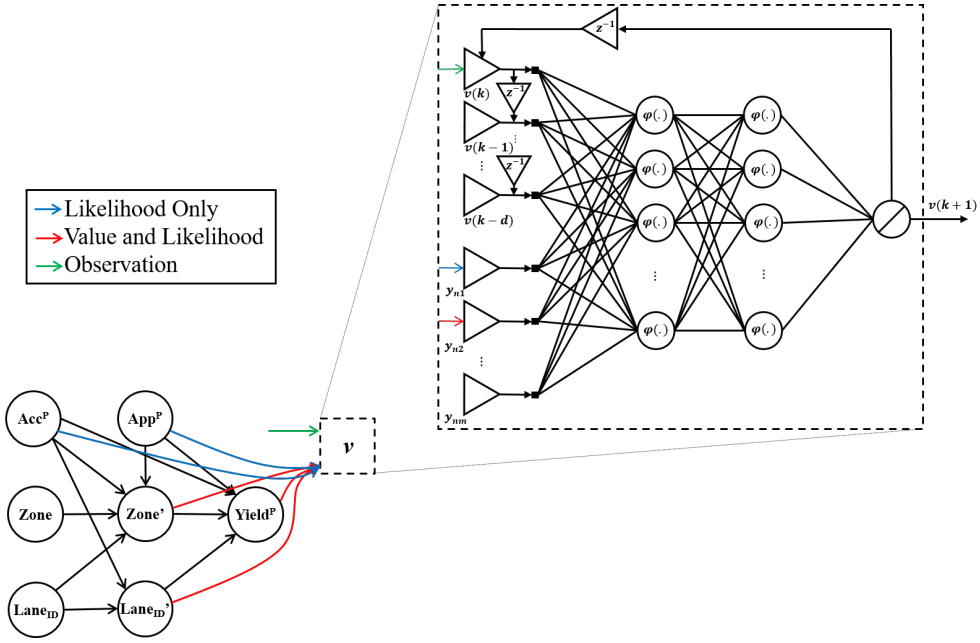



Figure 3.7: DBN-RNN combination schematic

Of course, the interface between the DBN and RNN should adopt a few changes. For variables with only two states (i.e. binary variables) DBN output to RNN input can be the conditional probability of one of the states knowing that the sum of a variable taking all of its states given the same conditions is 1 [32].

$$\sum_{A} P(a|X) = 1 \tag{3.1}$$

For variables with more than two states, the inputs can be a combination of the mean, and the likelihood of the variable distribution class with the highest conditional probability. The resulting array replaces $y_{n1}, ..., y_{nm}$ in Figure 3.7. Figure 3.7 Illustrates the complete schematic of the DBN-RNN interface.

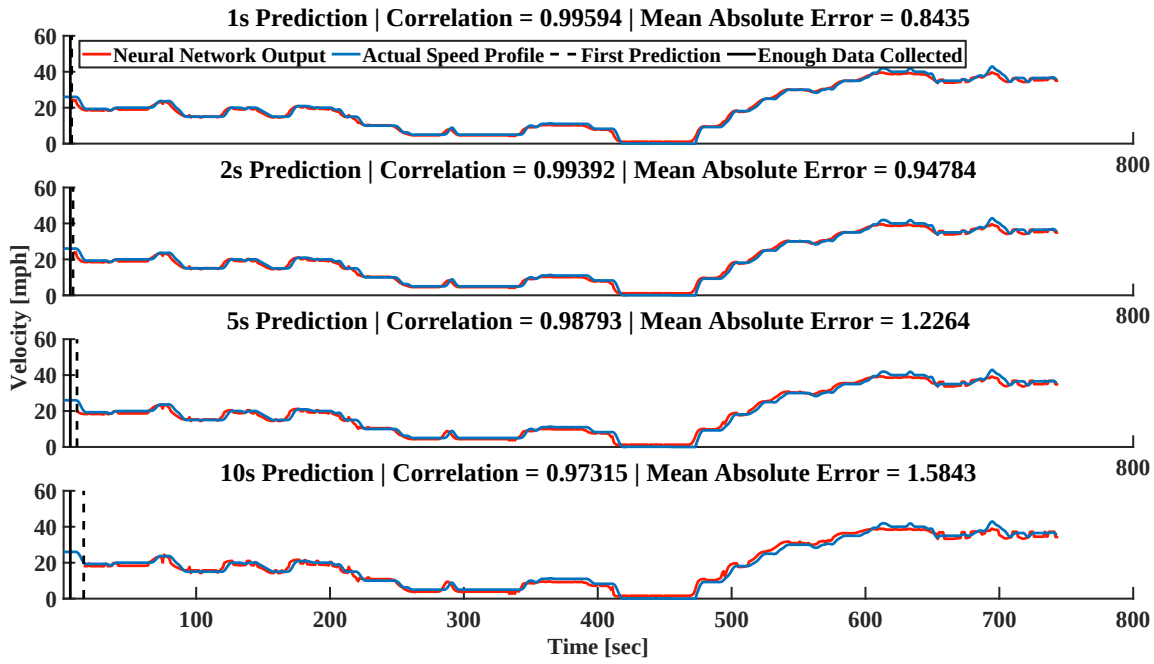
### 3.2.2 Results

The results yielded by the prediction network are the product of DBN trained with the entire network using EM-Algorithm and the RNN with 12807 sample points due to computational restrictions. In order to encompass as much variety in such limited size as possible, a search was carried out on the dataset to make sure the lowest, and the highest mean velocity and acceleration, and also the lowest, and the highest absolute velocity and acceleration are included. In addition to that, sets with and without full stops were picked to include rush hour and stopped traffic scenarios. The rest of the scenarios were chosen randomly using a roulette wheel selection weighted by data size. Choosing the specific scenarios also had a lower limit on size restriction. In total, 30 speed profiles were extracted from the dataset for training. After the training was over, scenarios were randomly picked from the entire dataset for testing. Typically, a 2-3 second prediction horizon would suffice for satisfactory control with MPCs, and that is where accuracy matters the most. However, for the sake of argument, all the results include the predicted speed profile for 1, 2, 5, and 10-second horizons. A common theme carried over in all the prediction results is that as the prediction horizon increases, accuracy is lost because errors become larger. However, it is evident that correlation (3.2) between prediction results and true speed profile, although reduced, remains relatively high (0.95 <).

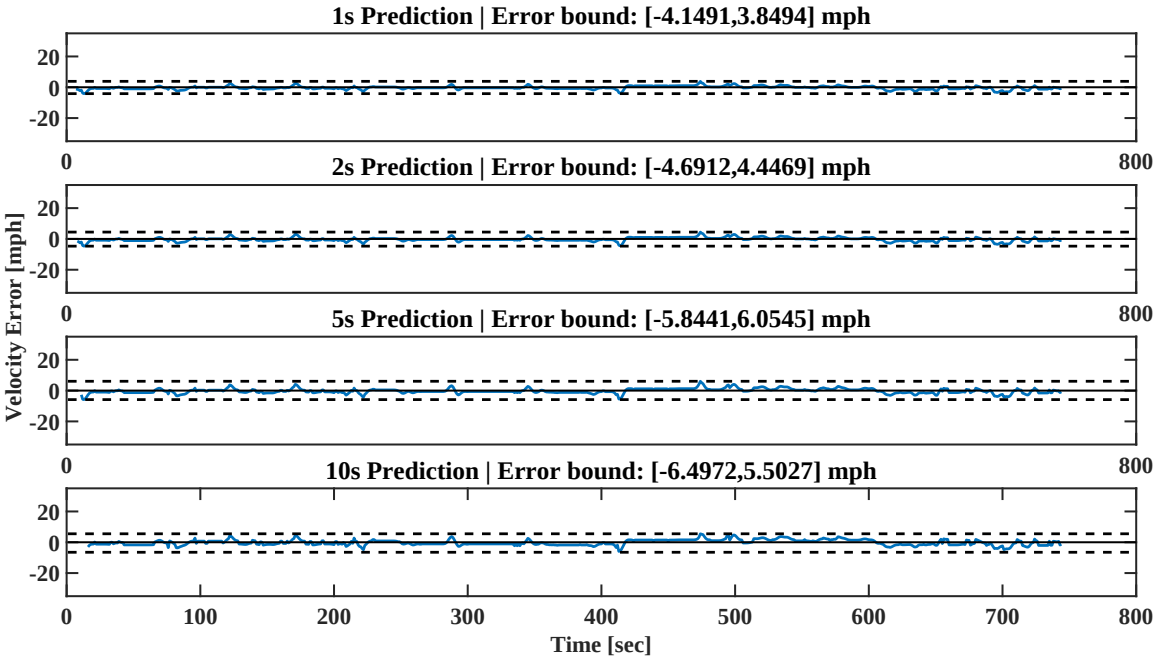
$$r_{xy} := \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.2}$$

Figure 3.8 illustrates the results yielded for rush hour traffic. As already mentioned, it is evident that the higher the prediction horizon goes, the mean absolute error grows. However, the trend is captured very well and one point that stacks up in favour of the results is that despite the training data being solely for highway driving, prediction results embody the stopped traffic. Expanding the training set should further improve the results as more samples are introduced to the network, which can be achieved with long term sequential off-line training, updating the weight matrix.

The general observations hold for normal driving scenario results illustrated in Figure 3.9, however, one main differences are setting this result apart from the rush hour traffic predictions; the data size is much smaller than the other two scenarios. Results suggest




(a) Network results



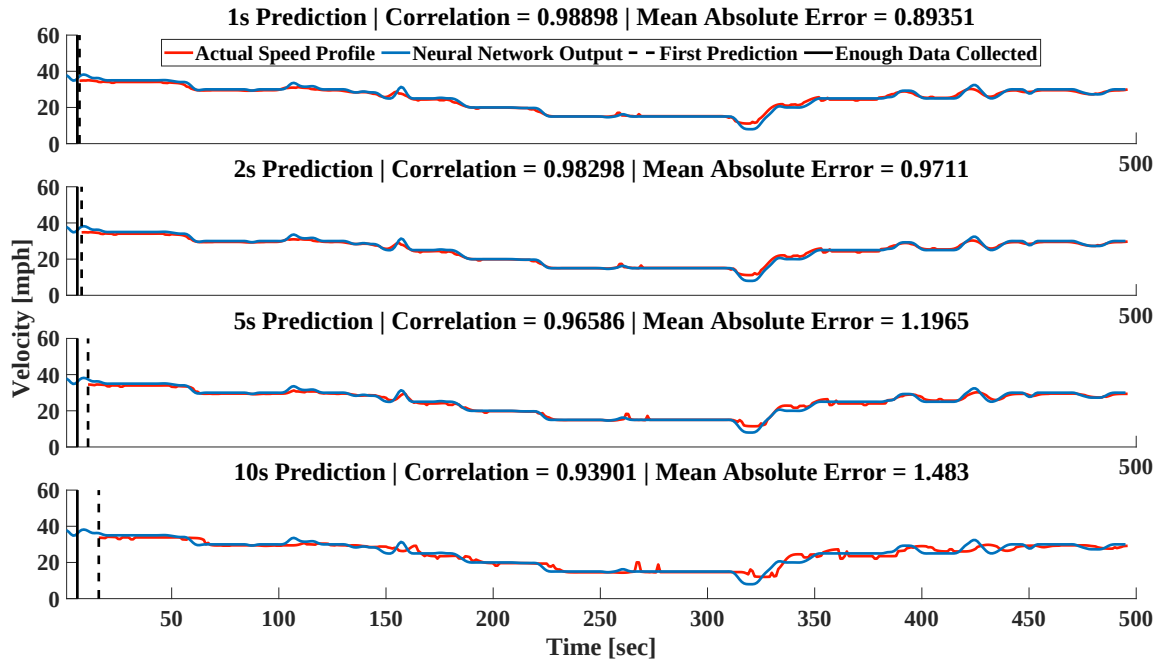

(b) errors

Figure 3.8: Prediction results for rush-hour traffic

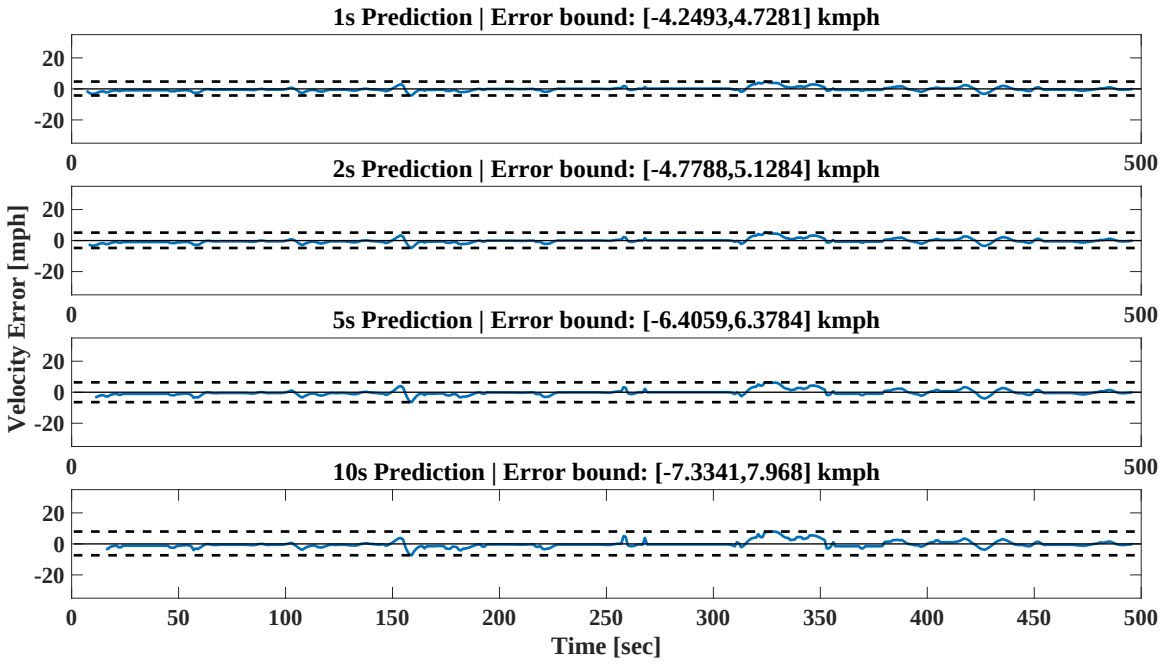
that there is a delay, or temporal difference, between the predictions and the actual speed profile which can be due to the changes in the companion variable inputs generated by the DBN, especially for $Lane_{ID}$ and $Zone$ variables, causing a different reaction from the RNN. However, up to 5-second horizon, the results are reliable.

Figure 3.10 represents a scenario in which the traffic comes to a complete stop for around 2 minutes, from second 400 to 520. Similar to rush hour traffic results, the stop period is very well captured, and the transient response seems promising. However, similar to normal driving scenario results, 10-second predictions display a temporal difference in the acceleration period after the full stop, which can be caused by the low acceleration rate at that instance. The accuracy is quickly recovered as the vehicle picks up pace.

Another main theme apparent in the results is that the errors in larger horizons are amplified versions of the errors in smaller horizons, which in turn makes the results further improvable by implementing filtering/smoothing methods. In one experiment with naïve Kalman filtering, speed prediction results of a primitive DBN topology was improved by a very large factor as the speed predictions exhibited large amplitude fluctuations, however the bestowed results do not involve any post-processing as they feature decent accuracy. However, for the sake of argument, filtered results are also presented in the forthcoming paragraphs. Compared to the one-dimensional car-following application considered in [8], the results of this study not only consider variations in velocity, but also can provide useful information such as lane change and yield probability which can be extremely useful for highway merging. The aforementioned loss of accuracy in larger horizons is also evident in the results of [19], however, unlike the results of [19] where the speed predictions naïvely diverge from the actual speed profile with the same rate, results of this study tend to exhibit a self-correcting behaviour as new cues are observed. Compared to the approach of [16], this study accounts for multi-lane merging and is able to predict a relatively accurate speed profile for an arbitrary positive integer horizon, although the yield/not-yield variable is shared between the two studies, the perspective on how to predict this variable is different. This study considers immediately measurable variables from the environment as opposed to time of arrival estimations; this allows for more accurate predictions, as well as the possibility of later development of the network topology as a result of the introduced versatility. As a general observation, most of the studies conducted on traffic prediction, look at the collective flow of the vehicles for long periods with low sampling rates and in large areas, aiming to predict congestions, similar to [33] or [34]. Similarly, [15] takes a long-term prediction strategy, which is justifiable for the applications. However, short-term predictions for individual vehicles become mission-critical and require attention to micro-details for any appropriate outcome from automated controllers. The reward of an accurate short-term vehicle behaviour prediction however, can directly improve the perfor-

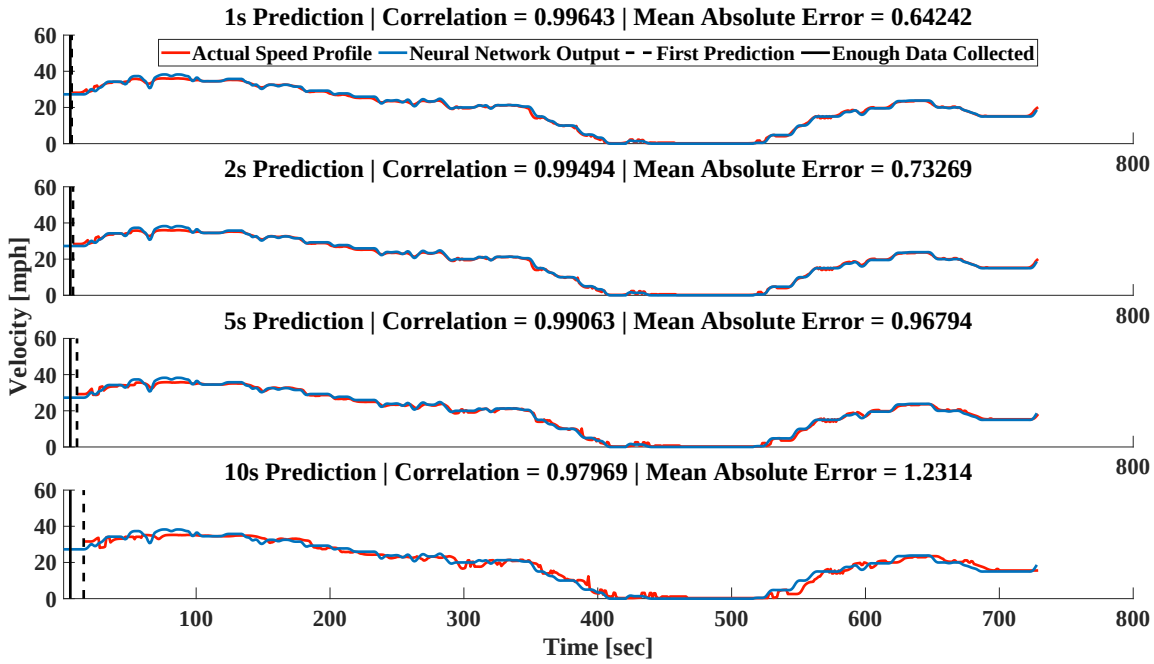



(a) Network results

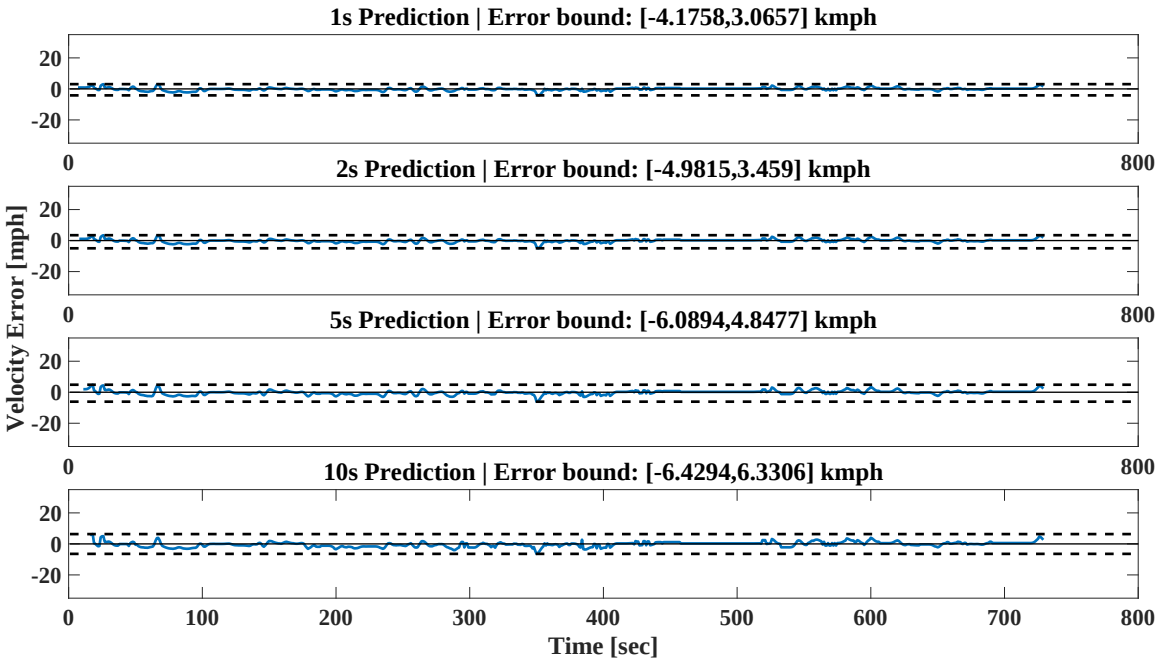



(b) errors

Figure 3.9: Prediction results for normal highway driving




(a) Network results




(b) errors

Figure 3.10: Prediction results for stopped traffic

mance of an autonomous or semi-autonomous vehicle. As mentioned above, [20] utilizes least-square parameter estimators for velocity prediction in an ACC developed by MPC. The performance of such controller would be certainly improved and would have allowed for more traffic scenarios given a more sophisticated prediction strategy, such as the one presented in this study. Similarly, in the controller defined in [7], predefined profiles can be replaced with a real-time predictor such as the one introduced in this study for further improved performance. Another example of a controller that can make use of traffic participant's state prediction is introduced in [28], where the *Multi-lane adaptive cruise controller (MLACC)* designed with a *non-linear model-predictive controller (NMPC)* assumes all the future states are fully known which is an unrealistic assumption. Similarly, in [35], the authors utilized a hybrid control method for vehicle platooning in a simulation where all the future actions are known and claim that utilizing future predictions of preceding vehicle's trajectory significantly enhances safety, driver comfort, traffic flow, vehicle emissions performance, and fuel economy.

As mentioned above, the trained DBN can also provide useful information to the driver as a stand-alone component, or connect with other types of controllers to ensure the safety of the manoeuvre. Table 3.2 is the direct result of EM-Algorithm and illustrates the distribution of yield probability given the conditions.

A close examination of Table 3.2 provides clues to better analyse the dataset and the methodology. First and foremost, this table contains 28 rows, while the complete set of conditions is supposed to contain 80 members (3.3).

$$\begin{aligned} |\{Conditions\}| &= |\{C\}| \times |\{P\}| \times |\{Z\}| \times |\{L\}| \\ &= 2 \times 2 \times 4 \times 5 \\ &= 80 \end{aligned} \tag{3.3}$$

However, because $Lane'_{ID}$ and $Zone'$ are not independent and have an overlap with each other. It would be physically impossible to have a condition where $Lane'_{ID} = 2$ and $Zone' = 1$ and so on. Set of permutations for $Lane'_{ID}$ and $Zone'_{ID}$ has therefore 7 members in total, combined with 2 states each for $Acc^P$ and $App^P$, there can be a total of 28 conditions for yield, all captured by the EM-Algorithm. Another seeming anomaly in the results comes from the coexistence of 'N/A' with 'yield' and '!yield'. It can be speculated that the reason arises from the fact that when defining the set values on the dataset, current $Lane_{ID}$ and $Zone$ were considered, however DBN, uses the next-step prediction of said variables, and therefore, a new uncertainty is introduced to the yield prediction. However, this newly introduced uncertainty is desirable, as it allows for prediction in yield, rather than explaining it. Considering that, by analysing the yielded results, it can be concluded that almost all the time, the traffic participant inside the highway yields to the

$Acc^P = C, App^P = P, Zone' = Z, Lane' = L$

| Row | Condition | !Yield | Yield | N/A |
|---|---|---|---|---|
| 01 | $C = 0, P = 0, Z = 1, L = 1$ | 0.0707 | 0.1304 | 0.7989 |
| 02 | $C = 0, P = 0, Z = 2, L = 1$ | 0.0202 | 0.0758 | 0.9040 |
| 03 | $C = 0, P = 0, Z = 3, L = 1$ | 0.0103 | 0.7105 | 0.2792 |
| 04 | $C = 0, P = 0, Z = 4, L = 2$ | $\approx 0$ | 0.7502 | 0.2498 |
| 05 | $C = 0, P = 0, Z = 4, L = R_{on}$ | 0 | 0.6381 | 0.3619 |
| 06 | $C = 0, P = 0, Z = 4, L = Aux$ | 0 | 0.4609 | 0.5391 |
| 07 | $C = 0, P = 0, Z = 4, L = R_{off}$ | 0 | 0.8467 | 0.1533 |
| 08 | $C = 0, P = 1, Z = 1, L = 1$ | 0.2490 | 0.2490 | 0.5019 |
| 09 | $C = 0, P = 1, Z = 2, L = 1$ | 0.1040 | 0.3564 | 0.5395 |
| 10 | $C = 0, P = 1, Z = 3, L = 1$ | 0.0099 | 0.7273 | 0.2628 |
| 11 | $C = 0, P = 1, Z = 4, L = 2$ | $\approx 0$ | 0.7483 | 0.2517 |
| 12 | $C = 0, P = 1, Z = 4, L = R_{on}$ | 0 | 0.7722 | 0.2278 |
| 13 | $C = 0, P = 1, Z = 4, L = Aux$ | 0 | 0.5440 | 0.4560 |
| 14 | $C = 0, P = 1, Z = 4, L = R_{off}$ | 0 | 0.8820 | 0.1180 |
| 15 | $C = 1, P = 0, Z = 1, L = 1$ | 0.1250 | 0 | 0.8750 |
| 16 | $C = 1, P = 0, Z = 2, L = 1$ | 0.0208 | 0.0812 | 0.8981 |
| 17 | $C = 1, P = 0, Z = 3, L = 1$ | 0.0123 | 0.6766 | 0.3111 |
| 18 | $C = 1, P = 0, Z = 4, L = 2$ | $\approx 0$ | 0.7125 | 0.2875 |
| 19 | $C = 1, P = 0, Z = 4, L = R_{on}$ | 0 | 0.6453 | 0.3547 |
| 20 | $C = 1, P = 0, Z = 4, L = Aux$ | 0 | 0.5270 | 0.4730 |
| 21 | $C = 1, P = 0, Z = 4, L = R_{off}$ | 0 | 0.8221 | 0.1779 |
| 22 | $C = 1, P = 1, Z = 1, L = 1$ | 0 | 0.3684 | 0.6316 |
| 23 | $C = 1, P = 1, Z = 2, L = 1$ | 0.1200 | 0.4176 | 0.4624 |
| 24 | $C = 1, P = 1, Z = 3, L = 1$ | 0.0089 | 0.7679 | 0.2232 |
| 25 | $C = 1, P = 1, Z = 4, L = 2$ | $\approx 0$ | 0.7811 | 0.2189 |
| 26 | $C = 1, P = 1, Z = 4, L = R_{on}$ | 0 | 0.8149 | 0.1851 |
| 27 | $C = 1, P = 1, Z = 4, L = Aux$ | 0 | 0.4414 | 0.5586 |
| 28 | $C = 1, P = 1, Z = 4, L = R_{off}$ | 0 | 0.9105 | 0.0895 |

Table 3.2: Yield probability table

merging traffic regardless of its lane-change behaviour (i.e. whether the yield is done by changing lane or slowing down). It is also possible to remove 'N/A' results after the fact by normalizing 'yield' and '!yield' linearly, or, to increase safety, weighting '!yield' by a certain factor. The higher the weight of '!yield' the more conservative the behaviour of the

autonomous merging vehicle.

As mentioned in the previously, the Kalman filter can be used to alleviate the jagged behaviour of prediction results by introducing a dynamic model to the system. To demonstrate, based on the simple kinematic model (3.4), state-space model (3.5) was created.

$$v(k+\Delta k) = v(k) + a(k)\Delta k \tag{3.4}$$

$$\begin{bmatrix} v(k+\Delta k) \\ a(k+\Delta k) \end{bmatrix} = \begin{bmatrix} 1 & \Delta k \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v(k) \\ a(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_v(k) \\ u_a(k) \end{bmatrix} \tag{3.5a}$$

$$Y(k) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v(k) \\ a(k) \end{bmatrix} \tag{3.5b}$$

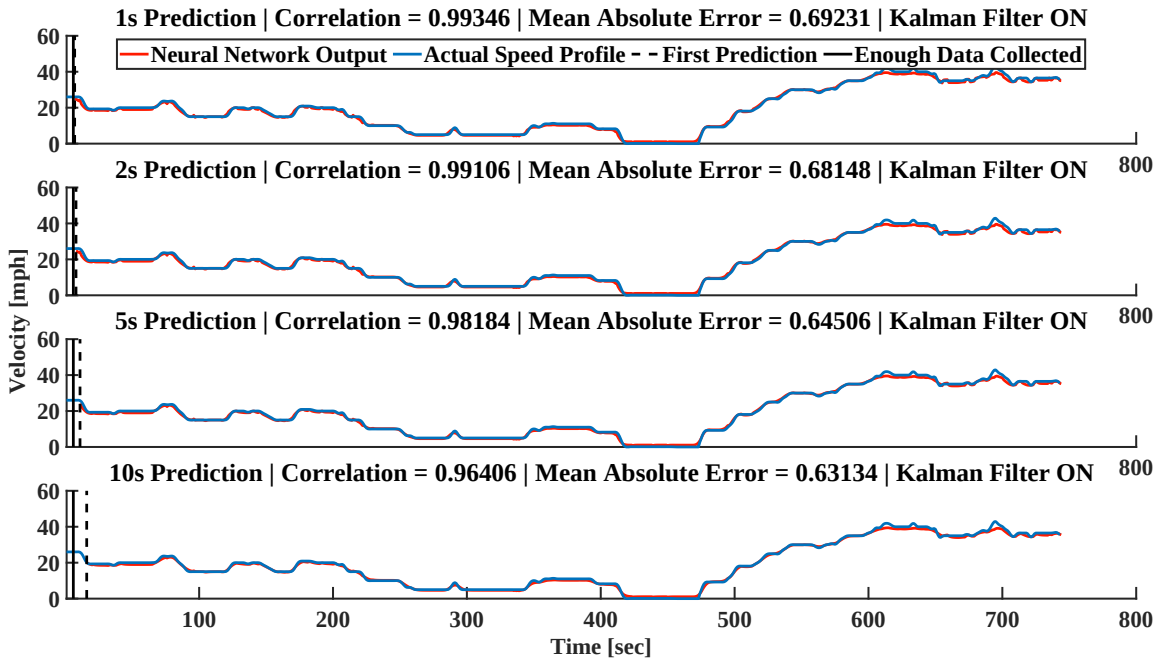
Then iteratively, input acceleration ($u_a(k)$) was calculated by differentiating neural network predicted velocity at the horizon and the predicted velocity preceding that by one step (3.6).

$$u_a(k) = \frac{v(horizon + k + \Delta k) - v(horizon + k)}{\Delta k} \tag{3.6}$$

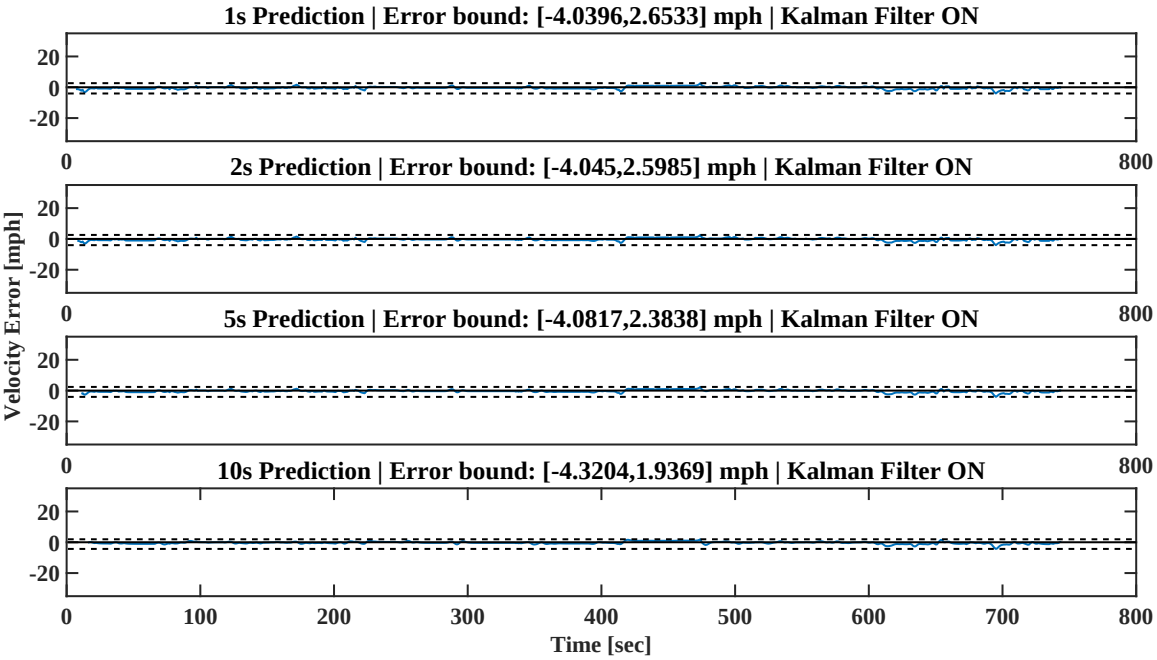
The sensitivity of each state is then determined by $R$ and $Q$ matrices to compensate for the lack of an accurate system model. The larger the coefficient of $Q$ matrix becomes, the less the Kalman filter has certainty over measurements, therefore the less the prediction step differs from the states at the current step. In other words, the filter has more inertia and resists the changes in states. By examining the prediction results without the Kalman filter, it is immediately apparent that shorter horizons have smaller jagged behaviour and are generally more accurate. Naturally, shorter horizons require a smaller $Q$ coefficient and longer horizons can benefit from a larger $Q$ coefficient. The results illustrated in Figure 3.11 are the outcome of $R$ and $Q$ shown below (3.7).

$$R = 1 \times I_2 \quad Q = \begin{cases} 0.02 & for\ 1s\ prediction \\ 0.10 & for\ 2s\ prediction \\ 0.60 & for\ 5s\ prediction \\ 4.00 & for\ 10s\ prediction \end{cases} \times I_2 \tag{3.7}$$

Considering Figure 3.11, using the same test data as Figure 3.8, it is evident that the Kalman filter is able to reduce the error and improve the overall performance of the predictions quite significantly, especially for longer horizons. In a more detailed comparison, while the correlation for 1s horizon is slightly hurt (most likely due to high $Q$ coefficient),




(a) Network Results




(b) Errors

Figure 3.11: Prediction results for stopped traffic with Kalman filter active

the errors are significantly lowered by the Kalman filter (60% lower mean absolute error for 10s horizon). Error bounds tell the same story as they are reduced by the Kalman filter across the board. This is especially beneficial as the error bounds were mostly affected by spontaneous spikes in errors rather than a consistent noise-type error. Kalman filter can remove the impulse errors exquisitely by giving inertia to the velocity predictions. Another area affected by the Kalman filter is along the horizontal axis for time/phase shifting. The DBN outputs include a limited number of states. The balance between the number of inputs from the environment and the number of inputs from the DBN going into the DBN will ultimately determine how the prediction results look like. However, for a large and sophisticated DBN, it is inevitable that DBN has more weight on the outputs, and as discussed above, it would not be the optimum solution to increase the number of environmental inputs. The effect of this DBN dominance comes in two forms. For one, as the string of DBN inputs switch states, there will be a sudden and abrupt change in the output of the RNN, which is the main cause of the 'jagged output'. For two, these abrupt changes usually translate into an advanced (i.e. ahead in time) response from the RNN causing a noticeable phase shift between the prediction results and the actual data. The effect of inertia introduced by the Kalman filter seems to reduce and diminish this behaviour, and although the $Q$ and $R$ coefficients are the result of trial and error in this study and only for demonstration purposes, a simple optimization algorithm can find the ideal coefficient for any use case and scenario. In summary, the benefits of 'inertia' in velocity predictions (and prediction for other continuous variables with the RNN) debuted by the introduction of Kalman filter is three-fold: First, it embodies the system's dynamic properties into the prediction results which is undeniably a welcome addition. It also filters the jagged output of RNN, due to dominant inputs from the DBN. In addition to that, it also compensates for the phase shift associated with dominant inputs from the DBN.

## 3.3 Scenario III: Intersection Driving [3]

Driving in an intersection, especially in such a chaotic environment alongside other human drivers, pedestrians, cyclists and other traffic participants is a rigorous and critical task. One of the biggest differences between human drivers and hypothetical fully autonomous vehicles would be the cues humans can interpret from other traffic participants, coming to them in the form of hand signals, body language, high beam lights and occasional

[3]This is an accepted manuscript awaiting publishing in the Proceedings of Canadian Society for Mechanical Engineering International Congress 2020 (CSME 2020), M.Zamani Abnili, N.L. Azad, "A New Data-Driven Approach For On-line Traffic Participant Behaviour Prediction at Intersections for Automated Driving"

activation of hazard lights for gratitude. It is simply not feasible to teach all of that to a machine, but the same machine, blinded to all these cues, is expected to yield better performance than human drivers. This expectation can only be realistic, relying on two basic principles.

1. The machine has more accurate measurements of the environment
2. The machine can execute control commands with more precision

However, given these two assumptions, the perception for a vehicle is still a complex challenge and a humongous task. In this study a simulation was assembled in Simulation of Urban Mobility (SUMO) [36], to collect the data needed for the data-driven techniques mentioned previously.

### 3.3.1 Simulation in SUMO

In SUMO, a controlled intersection (i.e. intersection controlled with traffic lights) environment was set up with two lanes on each side of each arm to incorporate lane changes as well as every possible traffic scenario, while maintaining manageable data size and flow densities. A total of 3600 vehicles were deployed in the environment taking random trips while conserving traffic rules. The traffic light signal schedule was chosen to be symmetric and simple without advanced left (as advanced left signal does not have any learning benefits). Figure 3.12 illustrates the SUMO environment. The data collected from the simulation was then fused with the map of the environment to create meaningful variables for the DBN.

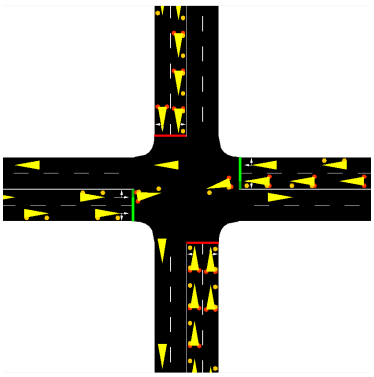


Figure 3.12: Intersection simulation environment

### 3.3.2 Dynamic Bayesian Network

The topology of the DBN must to follow a few basic rules:

- The source nodes (nodes with in-degree of 0) must be reserved for variables that can be directly measured from the environment. In other words, source nodes act as inputs.
- The complexity of the problem, for each node, scales exponentially with the number of parents and the number of states each parent can take. Therefore one node must not have too many parents.
- The child-parent dependency must be conserved. Each node should have either a direct or an indirect correlation with its neighbours.

Considering the rules mentioned above, DBN variables were established as declared in table 3.3. To create these variables, the only data collected from the simulation were position and traffic light signals, to be fed directly to the DBN, as well as velocity which is a continuous variable and was aimed to be handled by the RNN. Figure 3.13 illustrates the topology of the proposed DBN. In this topology, the status of the traffic light, the presence of vehicles in the vicinity, current lane, current section and whether the vehicle is approaching the intersection or departing from it are considered as inputs. Practically, these inputs can reliably be extracted from the environment, especially if V2X communications are considered.

To make predictions for continuous variables, the DBN outputs can be fed into the RNN, to complement the variable's history and add some context. However, was it not for the motion dynamics variables, the DBN would be sufficient to predict the states of any traffic participant, but because velocity is relative and depends on the states of the ego-vehicle as well as the traffic participant, by some means, both of these value sets must be exposed to the RNN. To achieve that, an interface must have been designed for this connection. There are three general ways that this interface can be implemented. The trade-off between these configurations revolves around the difficulty to prepare the data for, training time and result reliability. Figure 3.14 demonstrates simplified schematics of the architecture. In this scenario the first configuration was selected where the data for the ego vehicle and traffic participants are processed with the same simple DBN topology in Figure 3.13, due to shorter training times, easier pre-processing for the data and uncomplicated implementation.

[4] "Prime" denotes a variable in next time slice

| ID | Variable | Set of states |
|---|---|---|
| 01 | $StoppedAtLine$ | {True, False} |
| 02 | $LeftTurnRequiresStop$ | {True, False} |
| 03 | $PathIntersects$ | {True, False} |
| 04 | $CarFront$ | {True, False} |
| 05 | $CarSide$ | {True, False} |
| 06 | $CarBehind$ | {True, False} |
| 07 | $ApproachingIntersection$ | {True, False} |
| 08 | $AccelerationFlag$ | {True, False} |
| 09 | $Lane$ | {1, 2, Intersection} |
| 10 | $LanePrime$[4] | {1, 2, Intersection} |
| 11 | $TrafficLightSignal$ | {Green, Amber, Red} |
| 12 | $TrafficSignalPrime$ | {Green, Amber, Red} |
| 13 | $CurrentSection$ | {West, North, South, East, Intersection} |
| 14 | $SectionPrime$ | {West, North, South, East, Intersection} |

Table 3.3: DBN variables

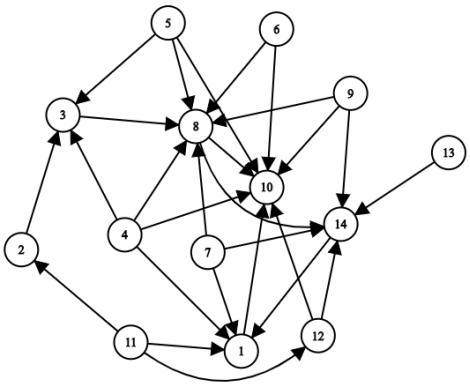



Figure 3.13: DBN topology

In addition to the macro connection scheme, DBN variables need to be singled out and selected to meet the RNN, as brute-forcing everything into the RNN will cause jagged

behaviour in the output. This jagged behaviour is because of the digital nature of the DBN outputs, especially when the distribution of states is not uniform across all the settings. The variables selected in this study are:

- $PathIntersection$
- $ApproachingIntersection$
- $AccelerationFlag$
- $TrafficLightSignalPrime$
- $SectionPrime$
- $LanePrime$

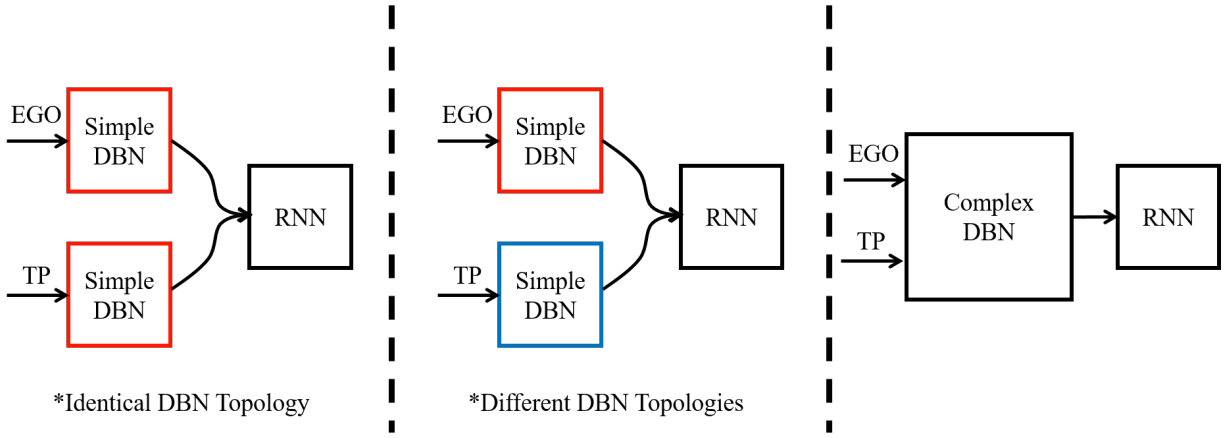



Figure 3.14: DBN-RNN interface

### 3.3.3 Recurrent Neural Network

In this scenario, the RNN is tasked with finding the velocity of a random traffic participant. While the DBN can quickly learn from very large datasets (in this study the DBN was trained with 480,000 sample points in a matter of seconds), the RNN takes much longer to train with the back-propagation algorithm, therefore only a small part of the data was chosen for training. This selection was based on data size and events (such as maximum and minimum velocities and maximum and minimum accelerations) as well as random

entries. The network structure is conical with 4 layers in total, with the number of nodes computed in each layer using (3.8), where $N$ is a constant for its subscript and $n$ is a counter. The history window ($d$) is chosen to be 6 seconds and sampled at 1Hz. The activation functions are sigmoid.

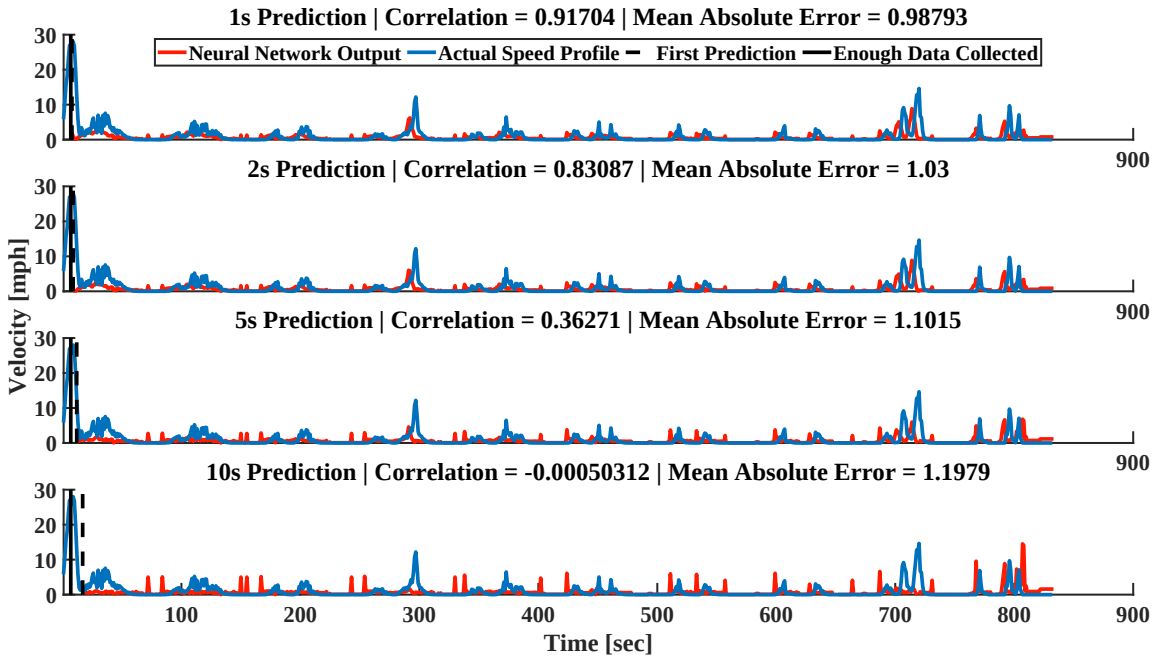
$$N_{nodes\ in\ layer} = (N_{ins} + N_{outs}) \\ * (N_{hidden\ layers} - n_{current\ layer} + 1) \tag{3.8}$$

### 3.3.4 Results

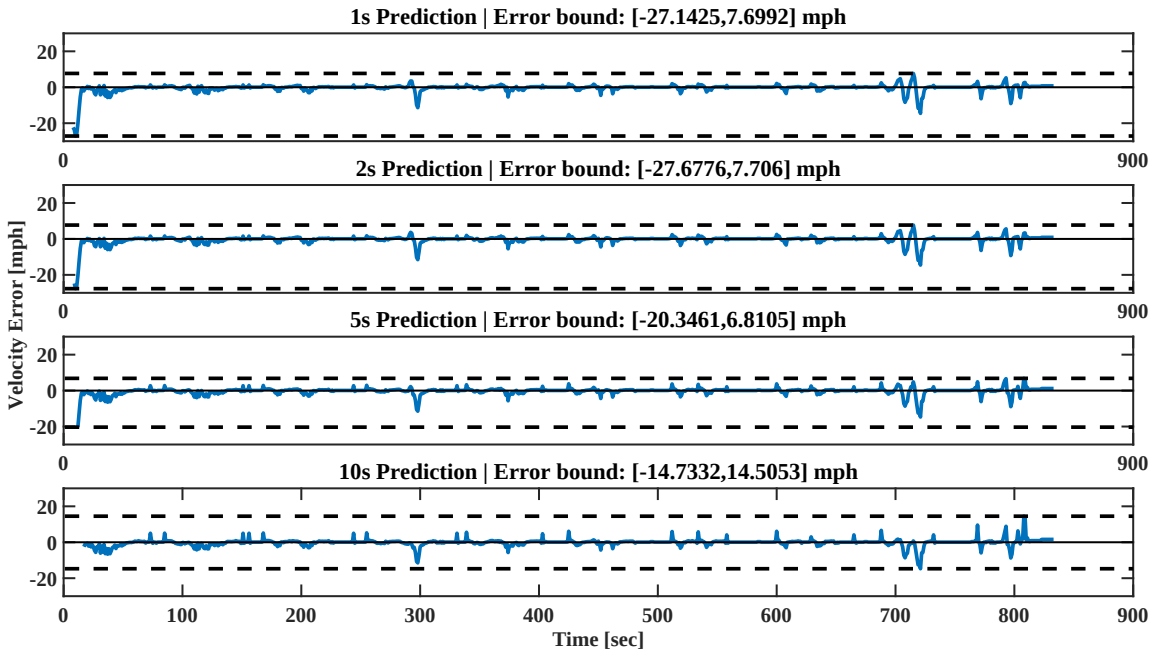
Velocity predictions for random vehicle pairs were established for 1, 2, 5, and 10-second horizons. A sample result is illustrated in Figure 3.15 without and in Figure 3.16 with Kalman filtering engaged. This specific drive-cycle is for a vehicle pair that coexisted in the simulation for a long period. This extended coincidence, alongside the multitude of full stops, signifies high traffic flow and multiple stops behind the traffic light. It is apparent that the prediction horizon has an inverse relationship with the prediction accuracy, and as the horizon extends, the errors become larger too. Figure 3.15 also exhibits error spikes in larger horizons which are legacy of the digital DBN inputs, these spikes are recognized to be mitigated by the Kalman filter in Figure 3.16. Results in Figure 3.16 appear to be following the trend of speed variations very well, however at some points, especially in the sudden peaks in the actual speed profile, such as the peak at 300 seconds, the scaling seems to be not quite exact, which can be attributed to the nature of neural networks. Compared to the results previously declared in [27], for highway merging where the velocities in the drive cycles are more gradually changing and have fewer stops, these results do not exhibit the same level of accuracy, however, this does not point to a futile fate for this approach in this application either. It is critical to keep in mind that intersection driving is by far and undoubtedly more complex than highway merging, and besides that, there are multiple amends for this issue. For instance, replacing the RNN with a more sophisticated method such as LSTM could be a proper adjustment to this approach. Furthermore, MPC, which are the prime target end-users for these predictions, tend to rely on 2-3 second prediction horizons which are adequately accurate.

## 3.4 Alternative Solutions (Extension of Scenario III)

There are enhanced methods that are direct upgrades to the 'vanilla' RNN that with extra gates and functions are designed to alleviate the vanishing, or exploding gradient problem.

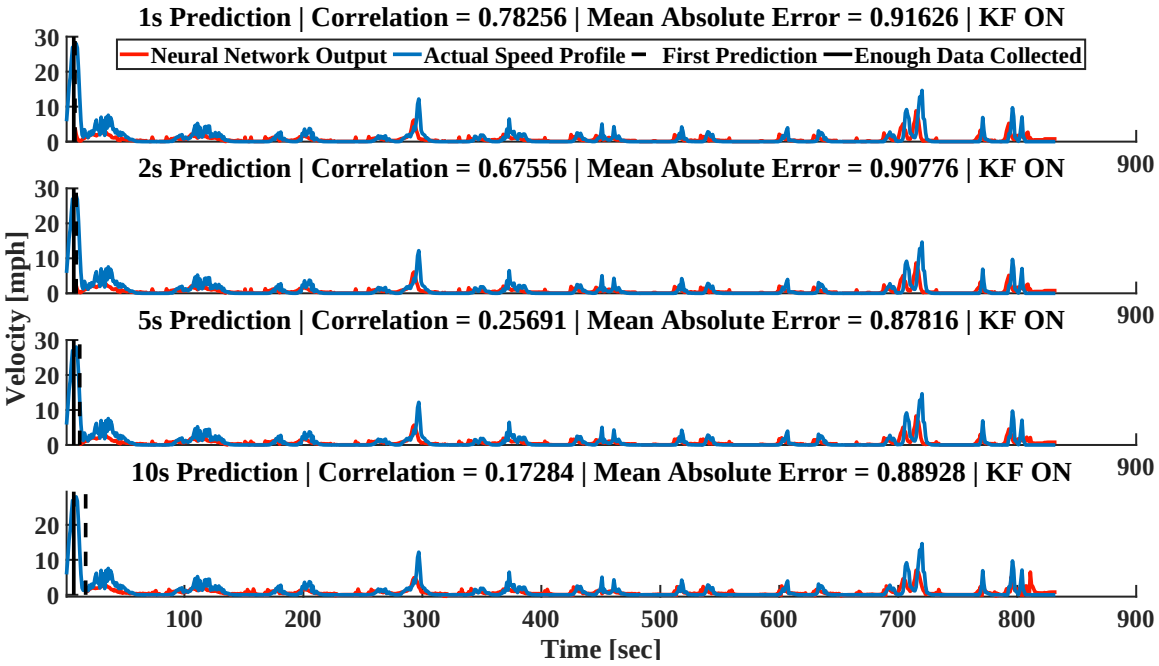



(a) Network results

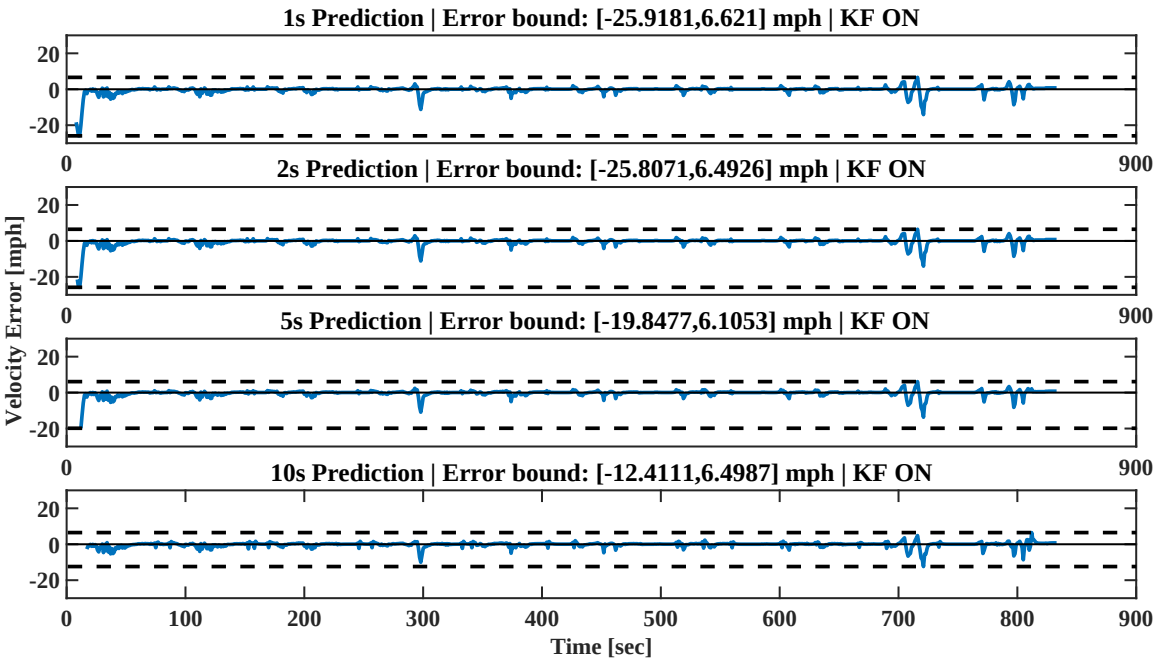



(b) Errors

Figure 3.15: Prediction results for random vehicle pair




(a) Network results




(b) Errors

Figure 3.16: Prediction results for random vehicle pair with Kalman Filtering

The vanishing, or exploding gradient problem is computational by nature and is a direct consequence of back propagation algorithm used to train the RNN with. Recalling (B.25, and B.9), some readers may notice that $\delta_{j,c}$, which is a term in (B.9), or the update to the weight matrix, is a function of error. As the training progresses, this error can become very small which in turn will make $\delta_{j,c}$ approach zero, or as training data becomes larger and with more variation, this term may become very large and approach infinity. This was especially a big issue in the past because of lower performance of computer hardware, rendering RNN and Back Propagation Algorithm (BP-Algorithm) completely useless in some applications. Now with computer hardware being much more sophisticated and capable the training RNN with BP-Algorithm is much more viable but the fundamental problem persists. Following the chain rule of training, the error signal in the front layers is exponentially smaller than the back layers, making training very slow in those layers.

The solution to this problem comes in many flavours. Starting with some of the first ones, introduced by Schmidhuber in [37] of an idea dubbed 'multi-level hierarchy of networks', the initial weight matrices for each layer is trained in an unsupervised fashion before training the RNN with BP-Algorithm, having the BP-Algorithm only do the fine-tuning. Having a good rough estimate of the weights before utilizing BP-Algorithm will allow for larger training rates while also preventing exploding gradient problem, and vanishing gradient problem at the same time. Another solution is utilizing residual networks (ResNets) which connects non-neighbouring layers, making the overall error larger.

At this time however, one cannot have "recurrent neural networks" and "vanishing gradient problem" in the same sentence and not talk about LSTM and GRUs. LSTMs and GRUs have special functions, or "gates", which are called: input, output, and forget gates for the case of LSTM, and update, and reset gates for GRU. These networks can be trained with a wide selection of optimizers ranging from stochastic gradient descent (SGD), to "Adaptive Gradient Methods with Dynamic Bound of Learning Rate (AdaBound)". This document will not dive in depths of the details and specifics of all the methods and techniques mentioned, however, a comparison between these methods and the results discussed in the previous sections would be a valid benchmark for the accuracy and performance of the approach proposed in this study.

Initially, as explained in the previous section, the alternative solutions were to replace the CVPM to benchmark the performance of the RNN. In total four different architectures were experimented with which were all laid out in the structure illustrated in the Figure 3.17. The GRU/LSTM were found to exhibit the highest accuracy when sandwiched between sets of fully connected layers, which were chosen to be 4.

Following the same structure, these networks had been experimented with using the same data and DBN topology used in Scenario III (results illustrated Figure 3.18 - 3.21).

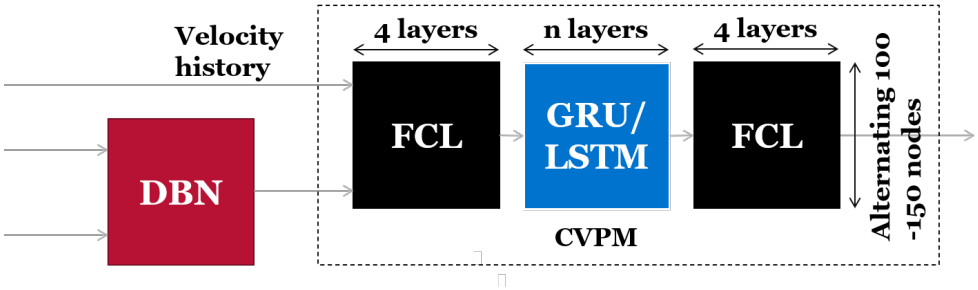



Figure 3.17: Alternative CVPM structure

| Method | Correlation at 5s | RMSE at 5s |
|---|---|---|
| RNN | 0.96921 | 0.05405 |
| Hybrid GRU | 0.93862 | 0.16569 |
| Hybrid LSTM | 0.85214 | 0.30139 |
| Hybrid Bi-LSTM | 0.85109 | 0.24268 |
| Hybrid LSTM+GRU | 0.74452 | 0.15472 |

Table 3.4: Results comparison

- Hybrid GRU network
- Hybrid LSTM network
- Hybrid Bi-Directional LSTM (Bi-LSTM) network
- Hybrid GRU-LSTM (alternating layers) network

Using the same test dataset as reference, the vanilla RNN results can be found in Figure 3.22. All the above networks were trained with the same data, settings, and properties.

Clearly the RNN has the superior results which can also be seen in Figure 3.23-3.24. Table 3.4 gives a numerical perspective of how these results compare.

All the above results were from the same global structure of DBN+CVPM. The superior results of RNN can seem counter-intuitive at first but it should be noted that the DBN stream was not sequentialized for the LSTM or GRU which is also the main reason why they had to be sandwiched between to sets of fully connected layers in the first place. However, there is more than one conclusion that can be drawn from the results; and that is the effect the DBN has on the performance of the CVPM. To observe this effect, the DBN stream can be removed from the LSTM inputs and have the trained network compete

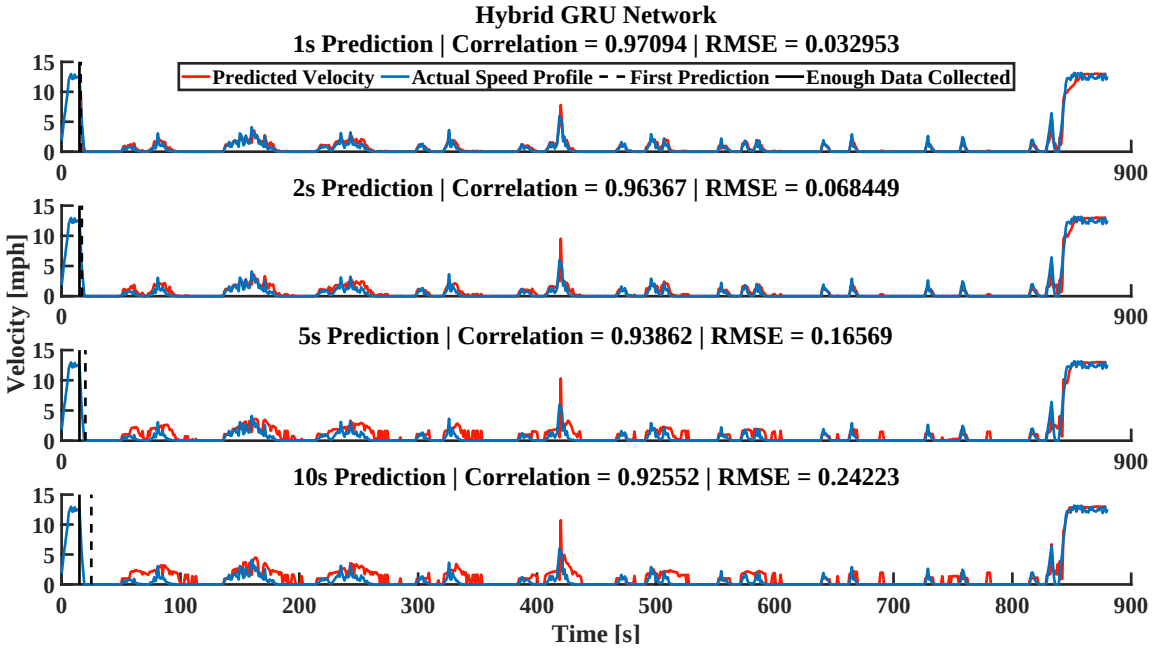



Figure 3.18: Hybrid GRU Network Results

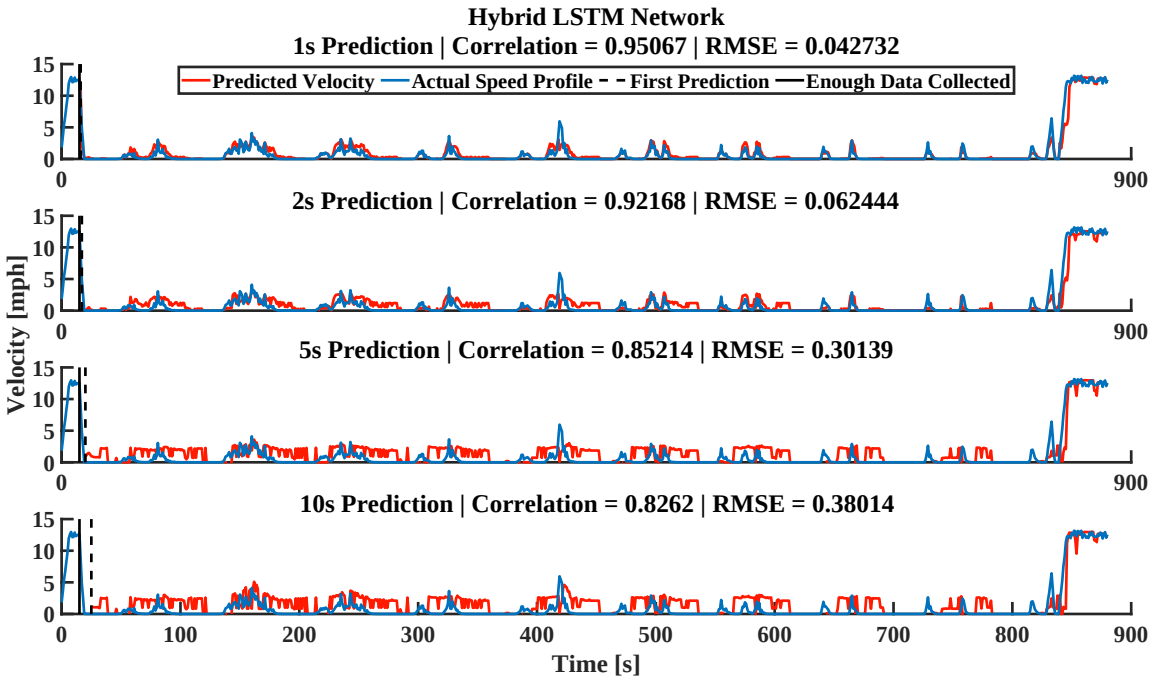



Figure 3.19: Hybrid LSTM Network Results

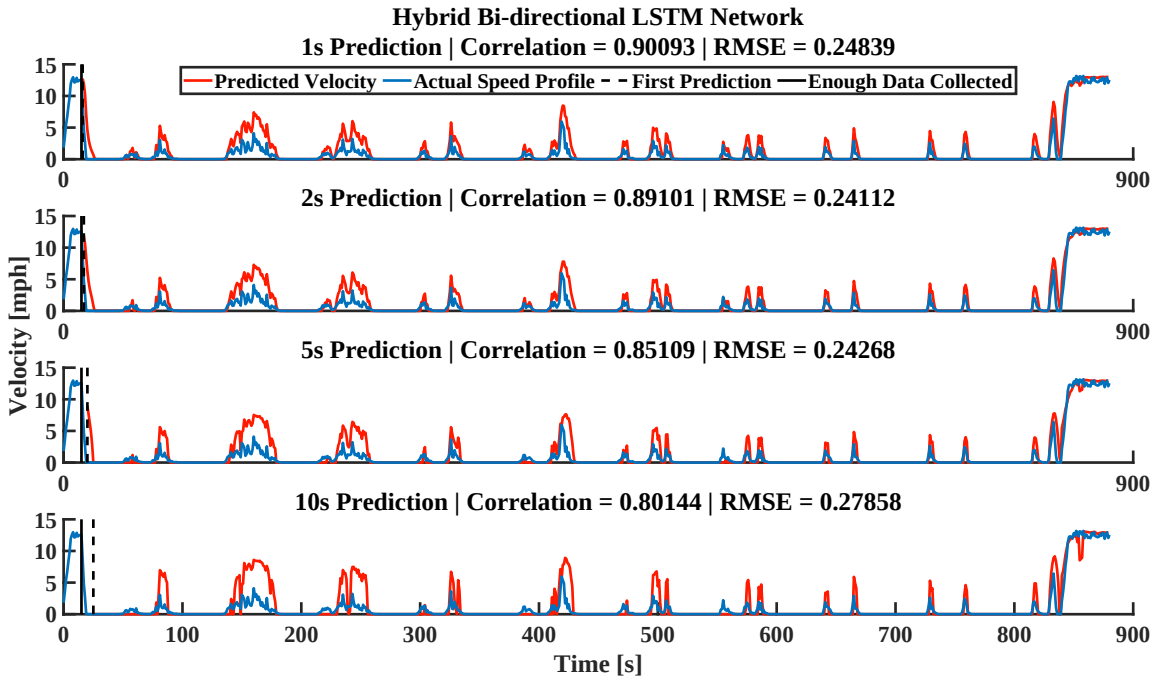



Figure 3.20: Hybrid Bi-LSTM Network Results

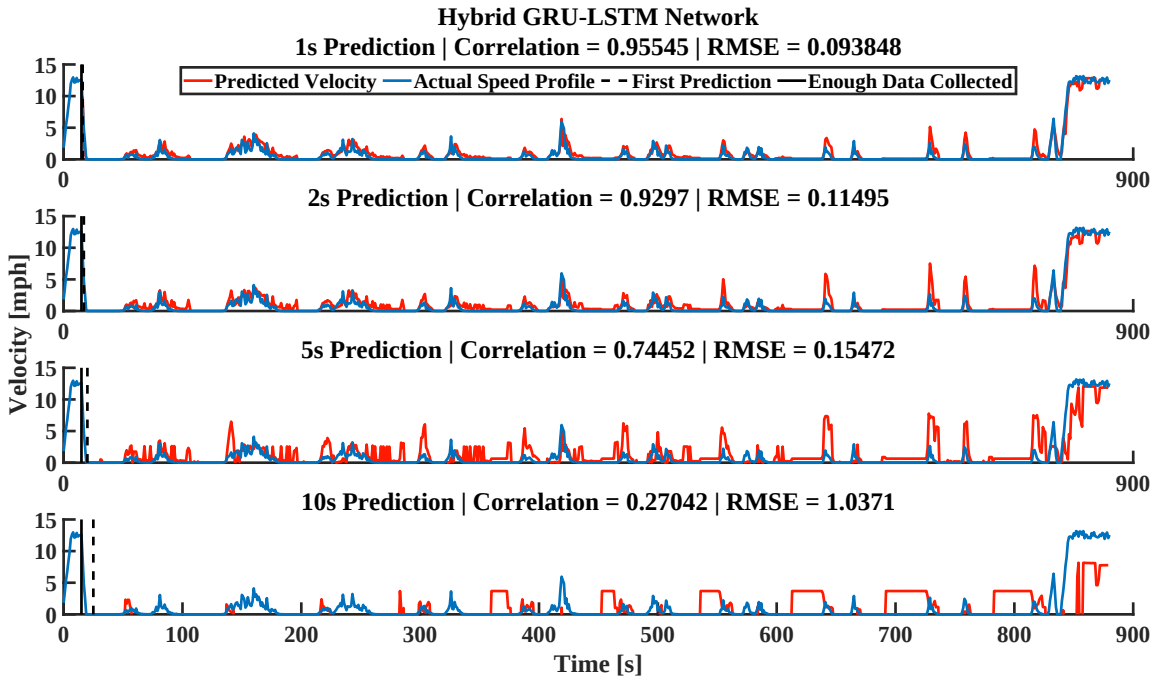



Figure 3.21: Hybrid LSTM+GRU Network Results

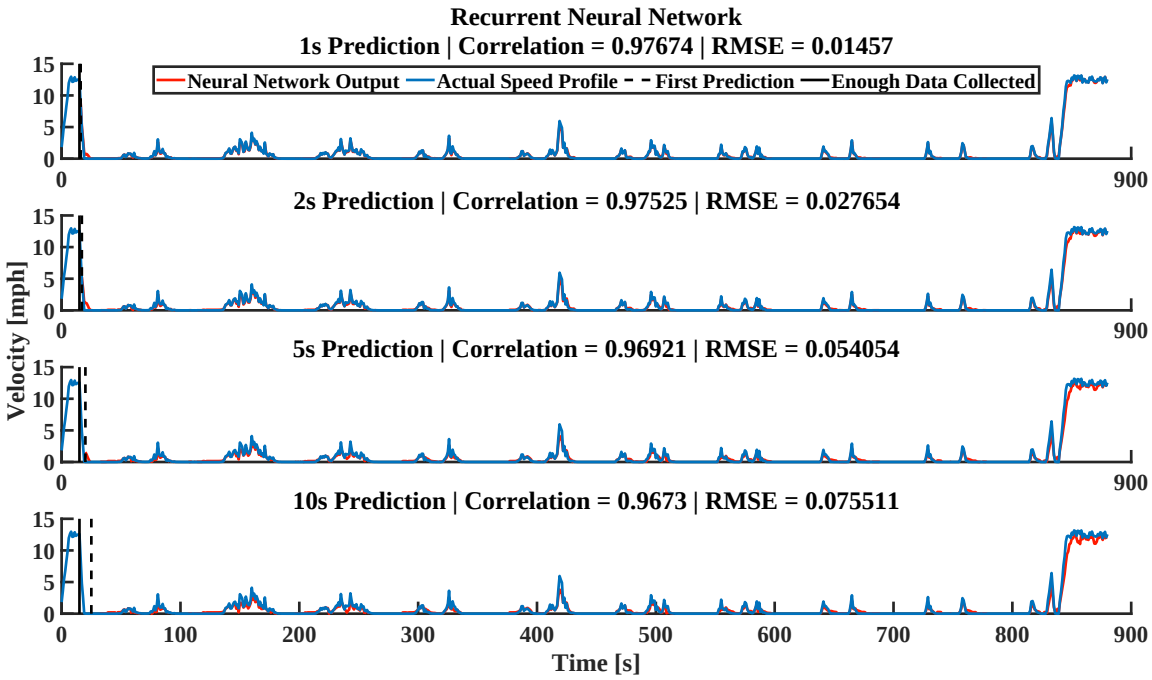



Figure 3.22: Reference RNN Results

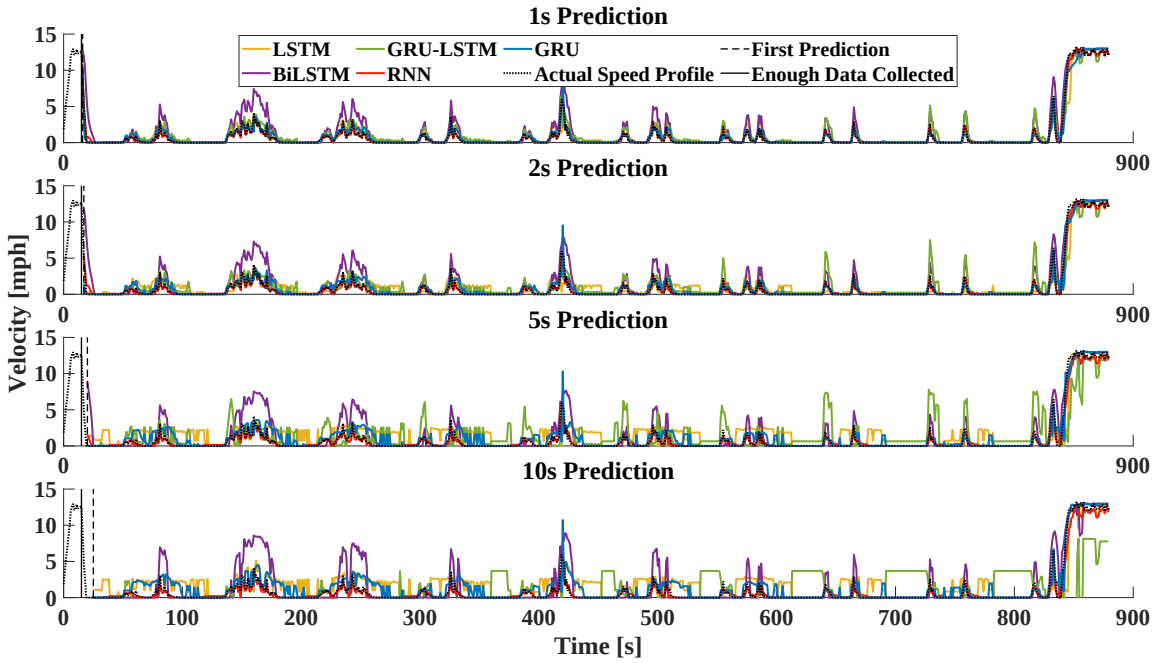



Figure 3.23: Stacked CVPM Comparison

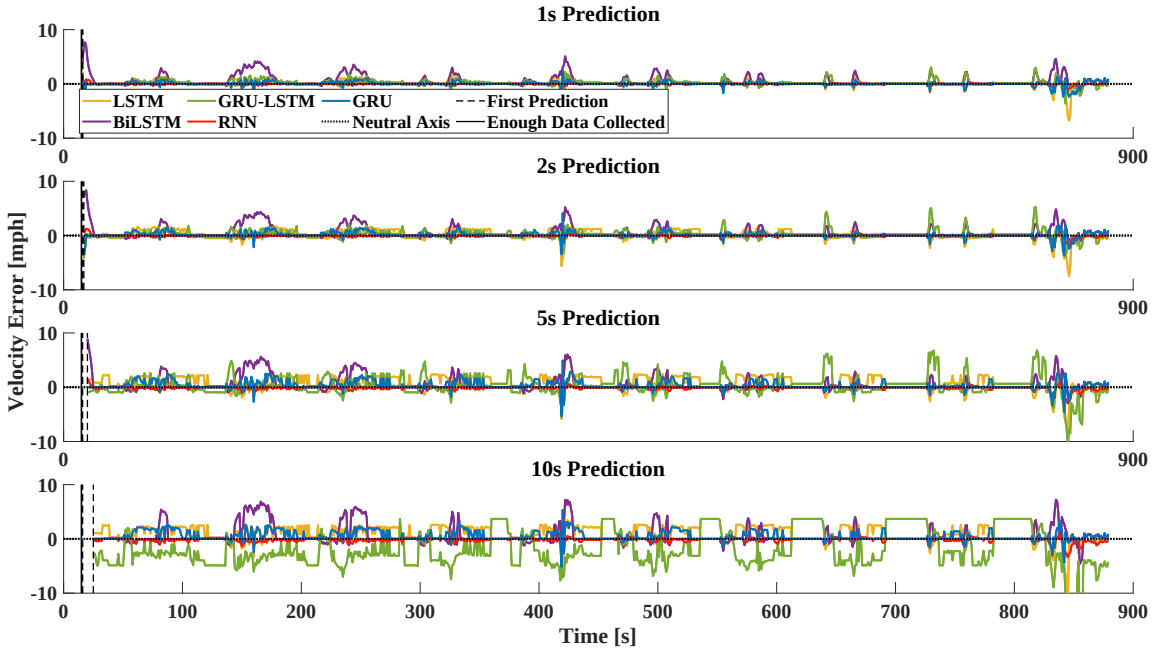



Figure 3.24: Stacked CVPM Errors Comparison

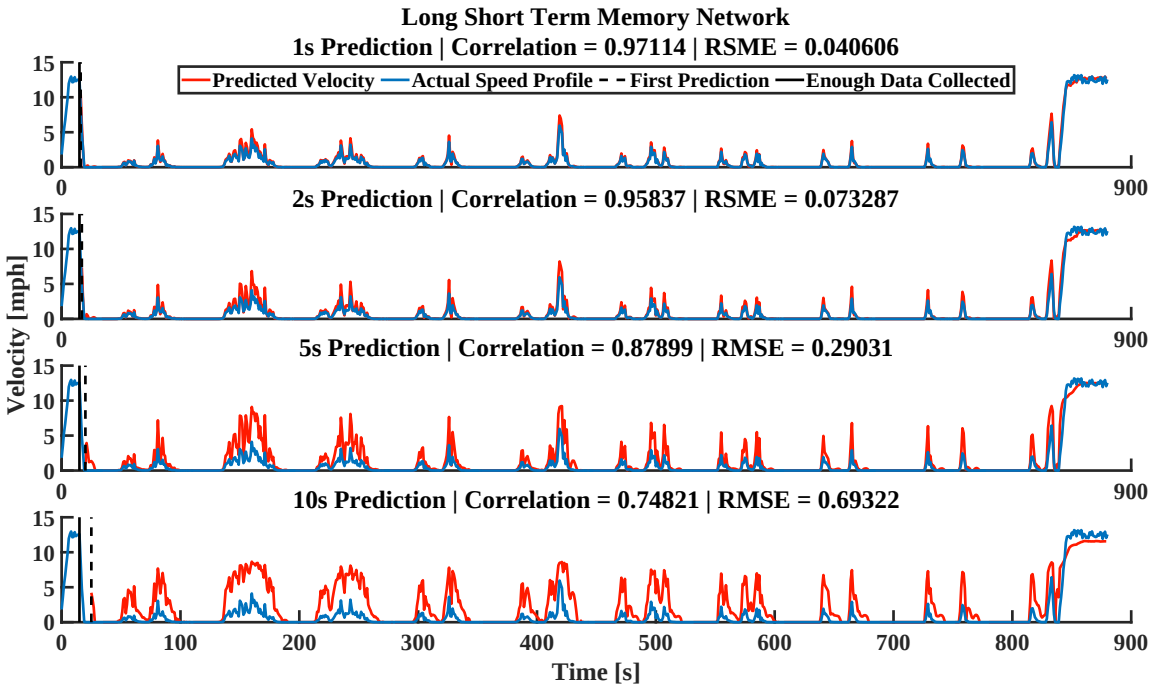



Figure 3.25: Solo LSTM CVPM

with the DBN+LSTM implementation which might have been less than ideal. Comparing Figure 3.19 with Figure 3.25, it is apparent that while effects of the non-sequentialized and digital-natured DBN inputs are removed from 3.25, the far more superior results of the Figure 3.22 indicate that:

1. DBN does in fact complement the RNN in terms of providing useful information
2. With tweaking the I/O stream, the alternative solutions could give very accurate results.

A possible solution is the revised approach illustrated in Figure 3.26 where a sequence of DBN stream is fed into a LSTM or GRU to classify a short-term speed change profile, which would then be used by another CVPM which can utilize that for its speed predictions. A by-product of this approach is that it provides more insight into what the CVPM is doing as the resemblance of the predicted velocity and the predicted velocity profile edge can be examined. However at the same time, some information is lost that could have been useful for velocity predictions in the CVPM. This can be compensated by reintroducing some variables into the direct DBN stream. Though overall, the computation time is expected

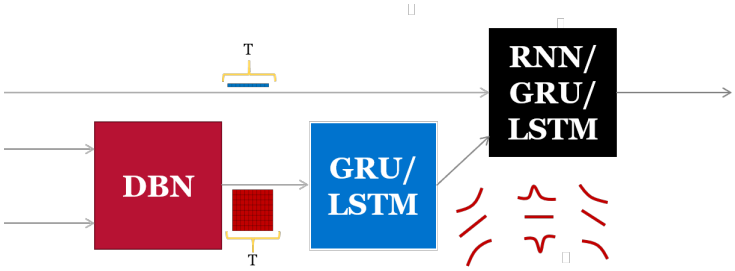



Figure 3.26: Revised approach

to be increased quite significantly which needs to be addressed. The satisfactory results of the DBN-RNN however, did not justify the extra steps needed for a possible improvement and the proposed method remains the same.

## 3.5 Scenario IV: Roundabout driving [5]

In this scenario, the strategy is applied to the case of roundabout driving which has been experiencing an emergence especially in growing urban areas as roundabouts offer a passive solution to traffic control which in turn makes them 'smarter' in controlling traffic. The lack of active components, for instance a traffic light signal, at roundabouts means that they are adaptive to the flow and temporal changes in traffic, but in turn poses a threat to inexperienced drivers who are not acquainted with roundabout driving, often causing a chain reaction in traffic accidents. This scenario is much more similar to Scenario II, than to Scenario III, in the sense that there is a priority relationship between the assumed ego-vehicle and other traffic participants. In the highway merging problem, the vehicle trying to merge into the highway is performing the more complex manoeuvre and can be focused on rather than having to consider vehicle pairs at all times as it was the case in intersection driving. In the roundabout driving, the vehicle trying to enter the roundabout is the point of focus as well, as once the vehicle has entered the roundabout, it has higher priority and can perform any manoeuvre without the need for complex decision making.

[5]This is an accepted manuscript published by Avestia in the Proceedings of the 7$^{th}$ International Conference on Control, Dynamic Systems, and Robotics (CDSR'20) on November 9th 2020, available online: http://doi.org/10.11159/cdsr20.155, M.Zamani Abnili, N.L. Azad, "Roundabout Situational Awareness for Automated Vehicles with Hybrid Machine Learning Approach"

### 3.5.1 Simulation in SUMO

SUMO does not support roundabouts natively, so the environment was created using a number of road sections connected via zippers in a circular shape. The increased number or zippers also acts as a lane-change preventing solution for inside the roundabout. Inside the considered roundabout, the priority for travellers has been increased so that the simulation is as close to reality as possible. Each arm extends for 500 meters as two lanes with a different traffic flow density to introduce variations into the traffic. A total of 16 flows were introduced at the end of every arm, one introducing one vehicle at every time step on the west side, one introducing one vehicle at every 2 time steps on the south side, one introducing one vehicle at every 3 time steps on the north side and one introducing one vehicle at every 4 time steps on the east side. In other words, the opposing sides have an equal total flow but placing the observer at each side would produce variation in the data. The flows have combined random trip destinations so that from every arm the flows generated will travel every four arms, but the sequence in which each vehicle is spawned into the simulation is random. A snapshot of the simulation environment is illustrated in Figure 3.27.

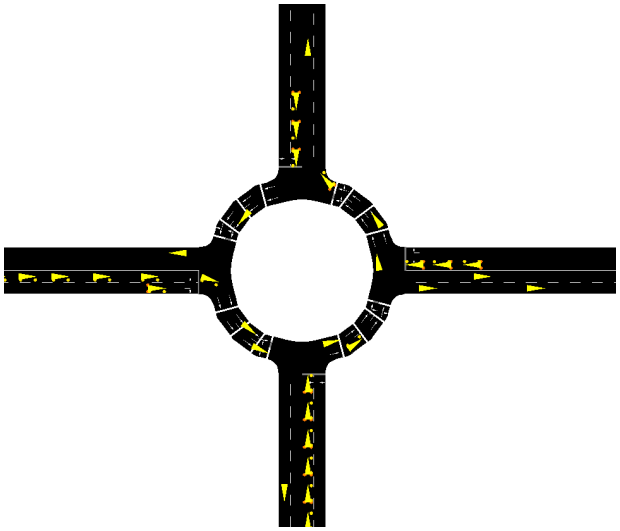
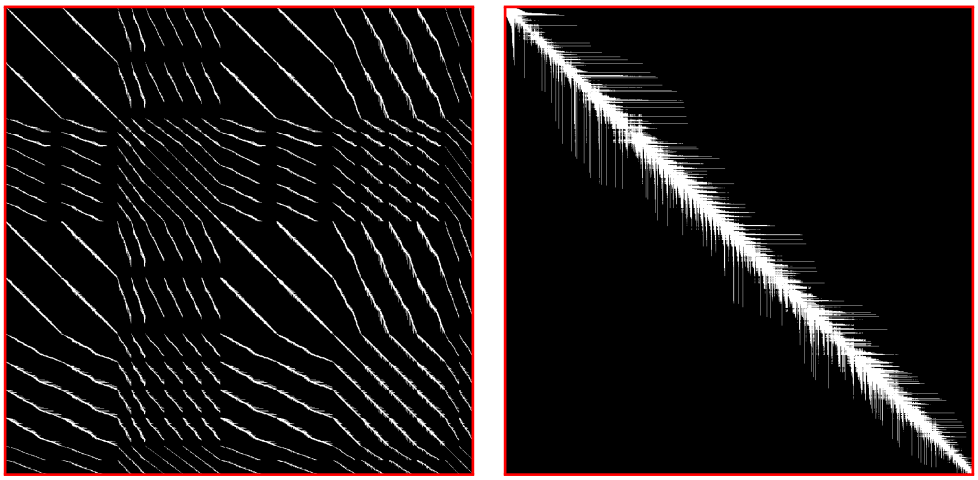


Figure 3.27: Roundabout simulation environment

### 3.5.2 Dynamic Bayesian Network Topology and Variables

The data extracted from SUMO has information about positions, speeds, and lanes of the 3000 vehicles in the simulation. The position information can be fused with the map of the environment to obtain relative positions of vehicles conserving the traffic rules. Due to the extreme size of the dataset, spectral clustering was performed on the set of vehicle pairs that were co-present in the simulation. In a 3000 $\times$ 3000 sparce grid of cells, the cells with data were rearranged to the closest configuration to a diagonal matrix. A visual representation of this rearrangement can be viewed in Figure 3.28 where each white pixel represents a cell with information and black background is formed by empty cells.



(a) Before clustering (b) After clustering

Figure 3.28: Visual representation of spectral clustering on the dataset

This rearrangement of data allows for much shorter processing times when extracting data for the DBN. The DBN topology is not guaranteed to be optimal and relies on the experts subjective judgement. In this case, the variables were chosen with two main objectives in mind:

1. Having the least pre-processing requirements
2. Being the most convenient to measure from the environment in a real driving scenario

After the spectral clustering, the following variables were extracted from the dataset by fusing the position, speed, and lane data with the roundabout layout. The variables, their states, and their layout can be found under Table 3.5.

| # | Variable Name | Variable States |
|---|---|---|
| 1 | Lane | {1,2} |
| 2 | Start Position | {E,N,W,S} |
| 3 | Entered RA [6] | {True,False} |
| 4 | Lane in RA | {1,2} |
| 5 | Destination | {E,N,W,S} |
| 6 | Inside RA | {True,False} |
| 7 | Stop to Enter | {True,False} |
| 8 | Lane Prime [7] | {1,2} |
| 9 | AccFlag [8] | {Negative, non-Negative} |
| 10 | AccFlag Prime | {Negative, non-Negative} |

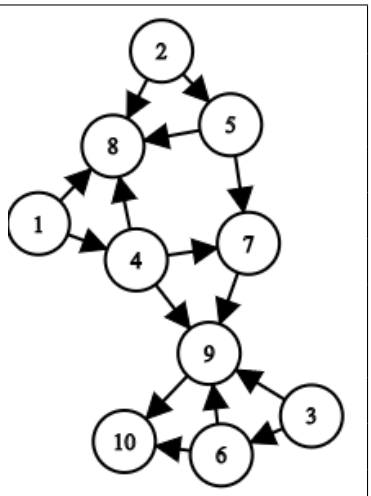



Table 3.5: Variables and the topology of the DBN used in Scenario IV

### 3.5.3 CVPM Implementation and DBN interfacing

This scenario continues the trend of previous sections and experiments with multiple CVPMs in combination with DBN, namely the vanilla RNN, LSTM, and GRU. The input-output for the three different implementations are the same and consist of a 12-second velocity history, complemented by the DBN stream. The DBN stream consists of the likelihood of one setting for binary variables, and the mean and the likelihood of the setting with maximum likelihood for non-binary variables. In this specific application, due to low number of DBN nodes, it was possible to feed all the variables into the CVPM to complement the velocity history. To keep it fair between the three methods, the number of layers and nodes were kept consistent throughout the trials, as well as training properties. One thing to keep in mind, however, is that in the case of complementary inputs, the DBN stream in this case, the closer the sequence predictor structure to a perceptron, the smoother the output and the higher the accuracy. This is validated by the results illustrated in the following figures. Also to keep the results fair, Kalman smoothing was not implemented in these results.

Stacked plot with the RNN, GRU, and the LSTM results are generated and presented in Figure 3.29 with the errors illustrated in Figure 3.30. The results for the RNN are far

[6] RA stands for "Roundabout"
[7] "Prime" refers to the variable in the next time-slice
[8] Acceleration Flag

superior than that of the GRU and LSTM in this specific application and that is beside quicker training times and turnaround times. Also, in this specific application, between the GRU and the LSTM, the LSTM has the upper hand in terms of accuracy.

As the vanilla RNN proved to be the most accurate among the methods, we focused on the RNN as the method of choice. Figure 3.29 illustrates the prediction results for a vehicle that spent a relatively short period in the simulation. The low average speed and frequent stops are markers that there was significant amount of traffic present during this trip, however the traffic has not caused the traffic to stop completely. That is as opposed to the trip taken in Figure 3.31 where the trip takes three times longer for the same distance travelled. Either result demonstrates very high prediction accuracy for both low speed traffic driving and following the transients. The complementary DBN stream has the effect on velocity prediction where there is clear distinction between low speed travel and complete stop, even for extended periods. The way DBN can achieve this is introducing a switch with many configurations each having a different weight that the RNN learns through back propagation algorithm. The error bounds for normal driving scenario for the RNN are [-0.71,0.94], [-0.97,1.47], [-1.46,2.75], and [-2.11,4.10], and in the case of heavy traffic driving are also measured at [-0.75,1.23], [-1.58,2.25], [-4.62,4.63], and [-6.64,6.74] m/s for 1, 2, 5, and 10 second predictions respectively. One notable item in these results is that low speed predictions are a lot more accurate than the final high-speed transients as the vehicle is jetting off the simulation environment. The reason for this phenomenon can be explained by analysing the dataset. In this dataset, vehicles that spent longer inside the simulation (due to heavier traffic), introduce more samples in the training. Out of these extra samples most, if not all of them, are zeros or very small values. That is because the travelled length is equal for all vehicles and that would be the area underneath the speed curve. The only scenario where these extra samples are not zero are for the cases where there is one or more lane-changes going into, or out of the roundabout. There are ways to mitigate this issue by introducing low traffic periods and merging multiple datasets for different flow densities. However, as some readers may have noticed, the main objective of a roundabout prediction strategy is to make predictions for a merging vehicle into the roundabout, and that is where the challenge lies, as vehicles inside the roundabout have higher priority and can exit at any time without hesitation. In other words, once a vehicle has merged into the roundabout the problem is solved for the most part, and to predict a vehicles motion after the roundabout, much simpler solutions can be implemented like the one introduced in [25].

Finally, the accuracy and validity of the results presented were measured by cross-correlations between actual data and the predicted speed profiles. These metrics for the RNN in the scenario presented in Figure 3.29 are shown in Table 3.6. As expected, the

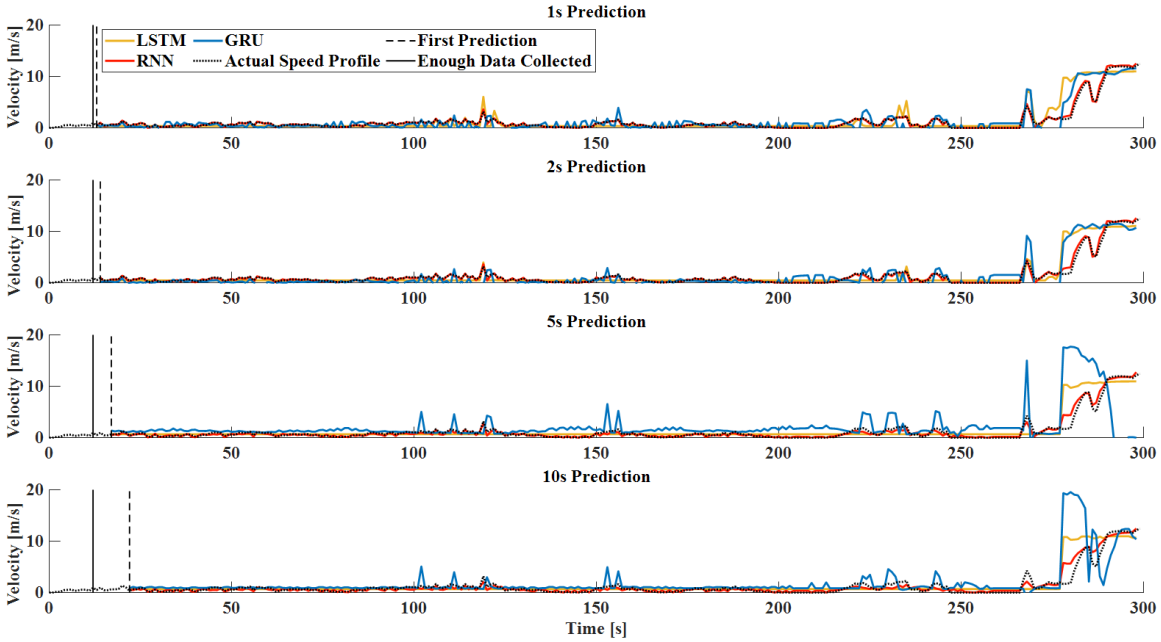



Figure 3.29: Stacked CVPM Comparison

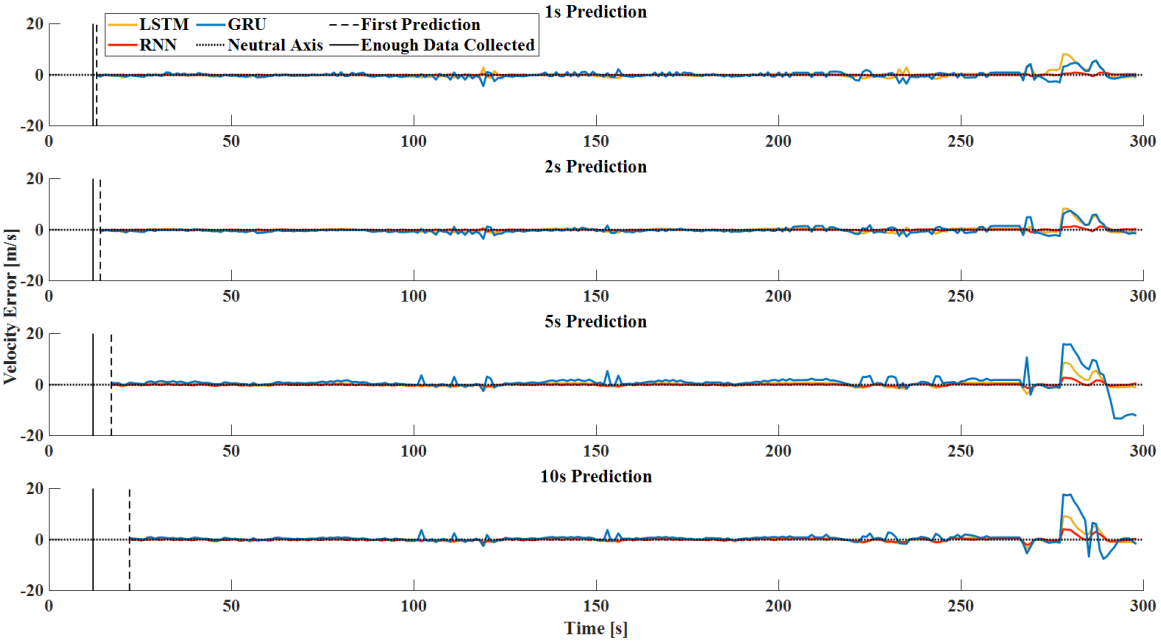



Figure 3.30: Stacked CVPM Errors Comparison

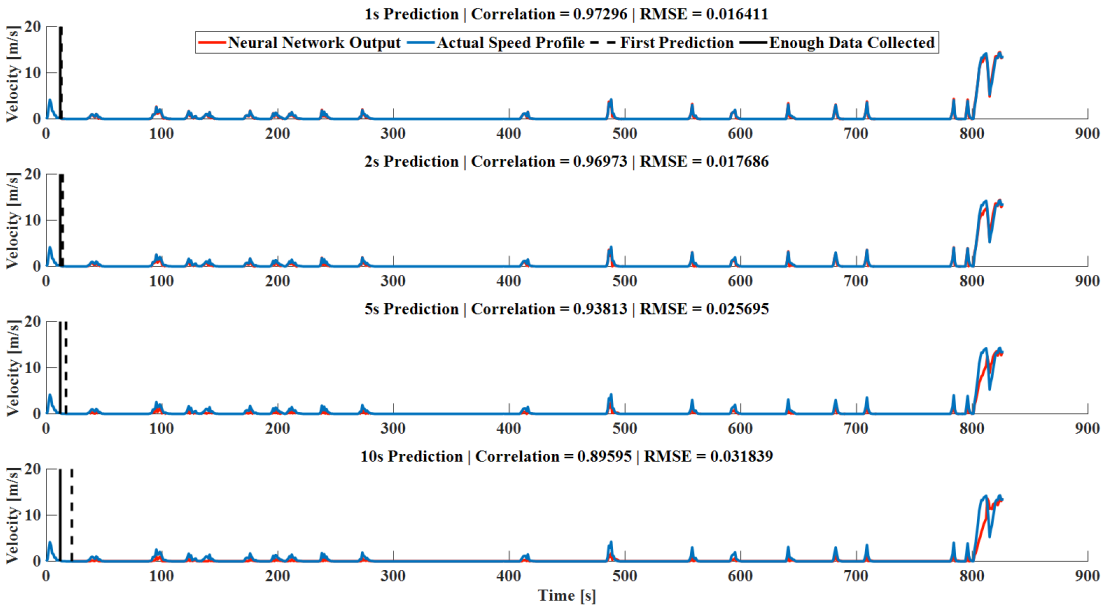



Figure 3.31: Roundabout results for heavy traffic

higher the prediction horizon, the lower the accuracy, but the results maintain a correlation greater than 0.895 and an RMSE less than 0.036.

| Prediction horizon | Correlation | RMSE |
|---|---|---|
| 1 second | 0.97641 | 0.029334 |
| 2 seconds | 0.97792 | 0.028453 |
| 5 seconds | 0.97661 | 0.029249 |
| 10 seconds | 0.96462 | 0.035995 |

Table 3.6: Accuracy metrics for RNN in scenario presented in Figure 3.29

# Chapter 4

# Development of a Scale-Car Platform for Test and Validation Purposes

A test platform with scaled cars was developed in the SHEVS for perception and control strategies validation purposes. The platform consists of a road network with a layout that was designed to encompass all of possible traffic scenarios, two fully instrumented vehicles, a driver in the loop set-up, and a global positioning system. The objective of developing this platform was to have a way to test and validate control and perception algorithms developed in SHEVS lab.

Figure 4.1: Scaled autonomous vehicle test platform developed in SHEVS lab

## 4.1 The City Layout

The main objective in the design of the city layout was to have as many driving locations as was possible to fit inside the available lab space. The design and implementation was done by a final year design project team (DJABA) working with SHEVS lab and the final layout includes (Figure 4.2):

- Roundabouts
- Two/four-lane roads
- Stop and yield signs
- Uncontrolled (can be virtually controlled) three/four-way intersections
- Auxiliary lanes to merge the traffic

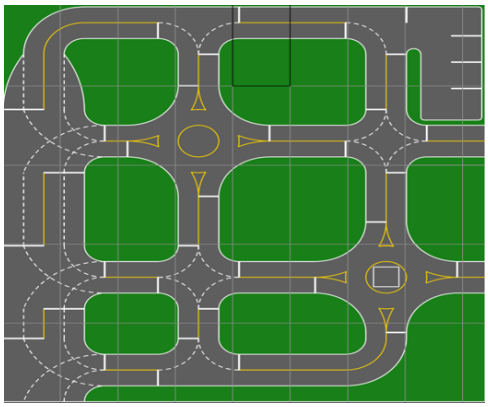


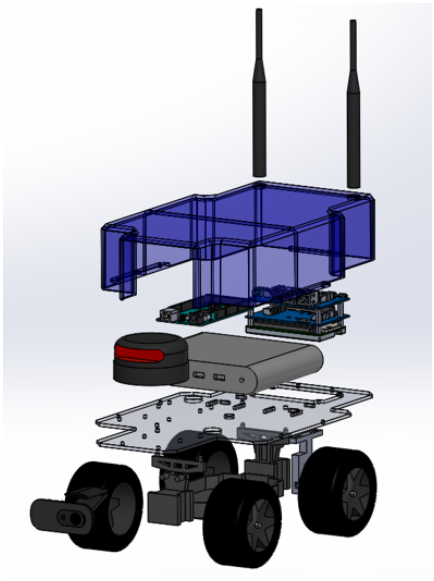
Figure 4.2: Scaled city layout

The city layout was printed on vinyl paper and was applied on panels so that the layout can be altered may there be the need to do so. This design was one of many designs that DJABA came up with and it was selected as it made the best use of the space.

## 4.2 Vehicles

The vehicles, also designed and implemented by the same FYDP team (DJABA), are heavily modified versions of RC hobby cars. The chassis and the vehicle propulsion are stock 1:18 ECX® Ruckus® 4WD Monster Truck. Since "monster trucks" are naturally larger with wider wheel base, effectively the end product has become 1:15 scaled standard car. On top of the chassis, sits a plexi-glass tray, housing the Nvidia Jetson™ TX2 as the high-level controller and the Arduino Mega 2560 as the actuator controller. The vehicles are equipped with LiDAR, Camera system, IMU, and wheel encoders. Figure 4.3a illustrates an exploded view of the vehicles and where each component goes (Note that this figure is missing the vehicle propulsion system, and steering servo). The shell was custom printed for these vehicles, as well as the encoder fork, and the inside teeth that would engage the encoder on the inside of the wheels. To support the added weight of the new components, the suspension was upgraded with new stiffer springs and to make everything fit inside the

small footprint, custom PCB boards were designed as Arduino shells and for the motor drivers. The actual hardware for the car can be seen in Figure 4.3b



(a) Exploded view of the vehicle components


(b) Actual hardware for one of the cars

## 4.3 Driver In The Loop

As mentioned before, with the development of any autonomous driving system, the main challenge is designing a system that can perform in presence of the uncertainties associated with humans. To include that in the test platform, driving simulation hardware was added that can take control of each one of the vehicles and then a user can drive the car using the camera feed transmitted from the vehicle to the screen in front of the driver. Figure 4.4 illustrates the hardware for the human-in-the-loop system. With the prediction algorithm, the thought process was that with a human-in-the-loop system the data collected for training can be a lot more realistic and with learning the nuances of human driving, it can produce much more accurate predictions which would be used as a validation method for the prediction strategy proposed in this study. The proof that this strategy works with real data can also be seen in the results of Scenario II which was using real data obtained from US-101 but it should also be noted that such data is not available for every driving scenario and location, and existing data almost never has the desired variables.

The driver-in-the-loop hardware has an acceleration pedal and a brake pedal. As some

Figure 4.4: Driver-in-the-loop system

readers may notice, especially those experienced with scaled hobby cars, is that these cars do not have braking system. At the same time, due to their relatively low weight and high friction, and surface adhesion, once the accelerator pedal is disengaged, the vehicle comes to an stop quite quickly. Regardless of that, to make sure there is as much similarity to real vehicles as possible, the brake pedal was programmed to switch the polarity of the motors and set the duty cycle to (4.1) where *bpp* is the brake pedal position percentage.

$$D = \lfloor \frac{v.bpp\%}{v_{max}} \rfloor \tag{4.1}$$

Also for research convenience, regardless of the active control scheme, the camera feed from both cars is always shown to the user and gives the ability to the user to hijack any of the cars at will.

## 4.4 Control Network

The control network was built on ROS with python codes running on the Linux nodes. For the sake of simplicity and fluidity, the main control algorithms were chosen to run on a desktop node, and the Jetson™ to simply relay the information to the Arduino actuator controller and transmitting camera feed to the main node. However it is important to note

that the Jetson™ itself is powerful enough to handle running most control algorithms. A set of 8 Vicon cameras would provide accurate location of each vehicle to the controller which may or may not be used by the algorithm but having ground truth is always a welcome addition. To make sure the collocated vehicle (i.e. the vehicle that is not controlled by a human driver) follows the trajectory produced by the control algorithms, a least squares system identification was performed on the vehicles to obtain a state-space model. The system parameter identification was performed with a few simplifying assumptions. The assumptions were regarding the system as linear and neglecting friction, also neglecting slips and slides. The justification was that past the static friction which introduces non-linearity to the system, for kinetic friction, the elements of the model will compensate for it. For the bicycle model below where $\omega$ is the yaw rate, $v_{lon}$ and $v_{lat}$ are the longitudinal and the lateral velocities, $\varphi$ is the steering angle and $u_a$ is the propulsion input.

$$\begin{bmatrix} \omega(k+1) \\ v_{lon}(k+1) \\ v_{lat}(k+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} \omega(k) \\ v_{lon}(k) \\ v_{lat}(k) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \times \begin{bmatrix} \varphi(k) \\ u_a(k) \end{bmatrix}$$

Obtaining the $A_{3\times3}$ and $B_{3\times2}$ matrices will provide the model relating the discrete states in the current time slice to the next. Feeding the plant a long out of sync sequence of step inputs for $\varphi$ and $u_a$ will provide enough data to extract the A and B matrices from. For a dataset of length N, it can be written:

$$\begin{bmatrix} \omega(2) & v_{lon}(2) & v_{lat}(2) \\ \omega(3) & v_{lon}(3) & v_{lat}(3) \\ \vdots & \vdots & \vdots \\ \omega(N) & v_{lon}(N) & v_{lat}(N) \end{bmatrix} = \begin{bmatrix} \omega(1) & v_{lon}(1) & v_{lat}(1) & \varphi(1) & u_a(1) \\ \omega(2) & v_{lon}(2) & v_{lat}(2) & \varphi(2) & u_a(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \omega(N-1) & v_{lon}(N-1) & v_{lat}(N-1) & \varphi(N-1) & u_a(N-1) \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \\ b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \end{bmatrix}$$

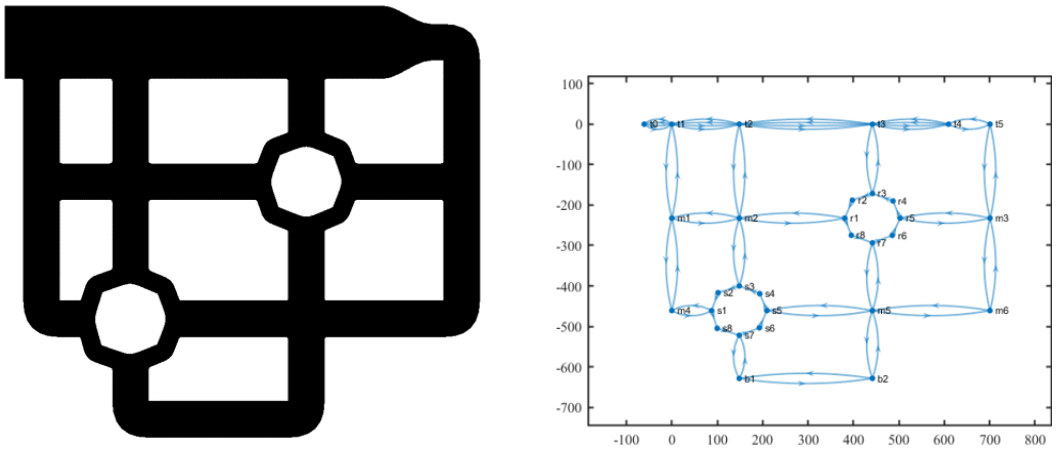
For the ease of notation, the matrices can be renamed to form the following equation:

$$\Omega = \Phi \times \Theta$$

In the equation above, $\Omega$ and $\Phi$ are known from the dataset and $\Theta$ contains all the model parameters, which can be singled out and estimated using the following equation:

$$\hat{\Theta} = (\Phi^T \Phi)^{-1} \Phi^T \Omega$$

The model was then used to obtain a linear-quadratic regulator (LQR) controller which would drive the actuators. The objectives for the controller were implicit with optional path following transformations in the feedback loop. Although a lot of non-linearity exists in the systems from friction, the LQR controller proved to work well. With that as a platform, a control algorithm for the vehicle in collocated control was developed by importing the map of the environment to SUMO, identifying where the vehicle is, then either receiving a user input for the destination or generating a random one. The algorithm would then find a path between the current location and the destination that would travel the least amount of nodes and feed the array of desired nodes to SUMO. SUMO would then generate the trajectory that the vehicle would have to travel to abide all the traffic laws. Figure 4.5 illustrates the city layout in SUMO and a directed graph representing the same environment. Figure 4.6 illustrates the control algorithm in action for a random initial position and a random destination.

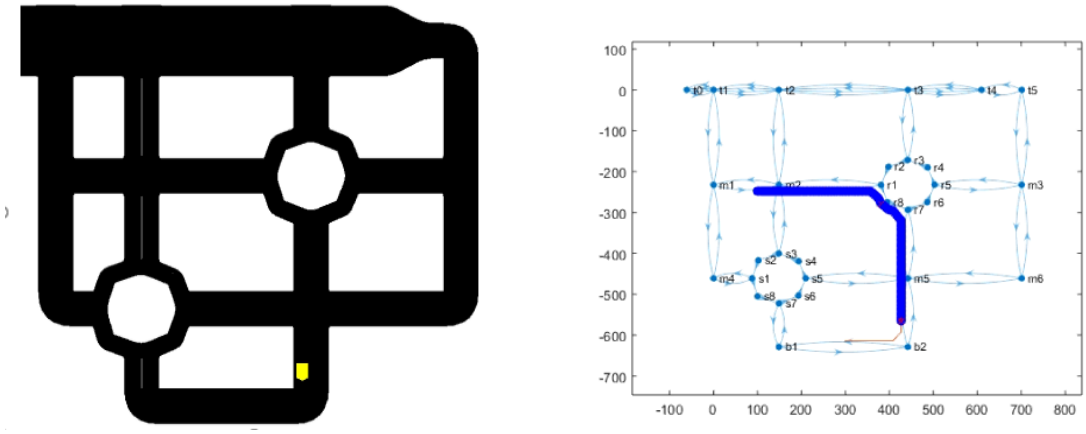



(a) Map of the environment in SUMO (b) Graphical representation

Figure 4.5: City layout

The benefit of using SUMO as a middle-ware, on top of introducing traffic laws into the control system, is that it makes augmenting the environment with virtual participants, and virtual constraints possible. The intersection can be controlled or uncontrolled, a path can be blocked, virtual traffic flow can be incorporated, etc., the possibilities are endless. It also provides a lot of potential for later expansion.

This setup was then utilized by another FYDP team (RoundRL) to test their reinforcement learning algorithm for roundabout driving. It was planned to have the test platform to validated the results of this study too by collecting data from the actual set-up and




(a) Map of the environment in SUMO (b) Graphical representation

Figure 4.6: Control algorithm in action

several human drivers and then employing the prediction strategy to predict the motion of any human driver driving the scaled cars. This however did not happen unfortunately because of the closures and lock downs related to COVID-19. With all the work that has been put into realization of this test platform, future generations of SHEVS lab researchers are strongly encouraged to utilize it to test their control or perception algorithms.

# Chapter 5

# Conclusion and Future Work

At the time of writing this thesis, autonomous driving is a very popular and trendy subject among research communities and tech industries alike. To recap from the introduction: "Autonomous driving promises affordable long-range transportation for the public by replacing the human workforce and minimizing energy consumption; Provides individuals with disability a means for personal conveyance, minimizes harmful by-products such as exhaust emissions in case of internal combustion engine-driven vehicles and decrease power plant waste in case of their electric counterparts. Most importantly, autonomous vehicles are expected to minimize traffic accidents and thereupon the property, and tragically in some cases life and limb, associated with such occurrences." Then there is no wonder as for the reason of this avid interest. That said, the challenges are also numerous and formidable. The main challenge arises from the fact that autonomous vehicles are not instantly replacing the human-driven ones and that translates to a transition phase where machines are supposed to operate among humans. Any robot operating among humans is required to meet very strict safety requirement and should be able to handle all sorts of uncertainties, let alone one that even when operated by humans, is responsible for so many tragic losses worldwide every year.

If we continue with the anthropomorphic theme of personifying machines, a machine's reactive brain, much like that of humans and animals, has two main sections: one responsible for understanding the environment, and the other for making decisions in accordance. Technically, for a machine these sections are called "perception" for the former and "controls" for the latter. The performance of such machine is as good as the weakest link it its brain chain. Whilst controls has seen a lot of research, and lots of progress, "perception" lagged behind, especially due to the computer processing technology not being up to par with artificial intelligence techniques.

For any driving scenario if there existed a solution that could robustly predict all the actions of traffic participants, and knew the traffic laws governing that scenario, the controller's job would then only be navigating collision-free trajectory, while minimizing fuel consumption and dynamic jerkiness. Although developing such controller is not an easy feat, such controllers already exist, thanks to years of research and development endeavours. The same cannot be said about perception strategies unfortunately.

In this study, a strategy for prediction of traffic participants' behaviour is proposed that can learn the rules that govern this behaviour by probabilistic inference. Due to complexity of the problem, and uncertainties associated with human behaviour, machine learning techniques were opted for, namely DBN and RNN. DBN is a powerful tool that can perform under uncertainty, partial observation, noisy inputs, and complex relationships between conjoint and dependent variables. At the same time, DBN cannot handle continuous variables and can be trained specific to one driving environment only. As for continuous variable prediction shortcoming, several CVPM techniques were experimented with, out of which vanilla RNN yielded superior results. For environment specificity, the solution is to enumerate profiles in which the DBN is trained. In this study this training was done for two different scenarios, highway merging, and a much more complex intersection driving, as well as roundabout driving. Adding a few more environment such as city (general), and highway (general), and then switching between the modes by intelligently identifying the driving scenario would make a complete package.

A pro and a con of DBN approach is it being an expert system. While there are methods to identify dependencies between variables empirically and quantitatively (e.g. by correlation analysis), expert knowledge is still required for constructing the topology. To justify this, one may think traffic rules are human contract anyway. As humans we learn it from experts, why should not the machine. While there is quite a substantial merit to this claim, it also makes this approach quite more difficult to implement and obtain good results from it as this "expert" needs to also think like a machine to be able to teach a machine. The advantage of DBN being an expert system however, is a natural one, and is the same reason why behaviourism in psychology is not perfect. As it is easier to train a dog to drool at the sound of a bell than it is to teach the same dog to dance, the expert knowledge is foundation for the algorithm to learn a specific dataset and can be thought of as its innate talent.

To be able to encapsulate the environment for the algorithm, an expert would construct the topology of the DBN with a few objectives in mind (3.3.2):

- The source nodes (nodes with in-degree of 0) must be reserved for variables that can be directly measured from the environment. In other words, source nodes act as inputs.

- The complexity of the problem, for each node, scales exponentially with the number of parents and the number of states each parent can take. Therefore one node must not have too many parents.
- The child-parent dependency must be conserved. Each node should have either a direct or an indirect correlation with its neighbours.

From there onwards, any continuous variable is switched to be handled by and RNN, that can observe a short history of that variable and make predictions for an arbitrary horizon. The overall schematic of this marriage can be summarized into Figure 5.1.

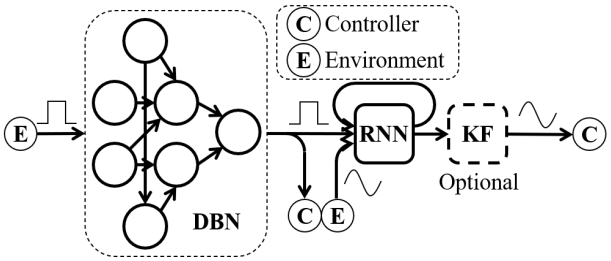



Figure 5.1: Components' combination schematics

While the proposed method does not solve the perception problem completely, the competent results yielded for highway merging (Scenario II) and intersection driving (Scenario III) promises a capable method that can complement controllers for improved performance. The areas for improvement are plentiful but can be classified into two main categories, accuracy and generalization.

In the case of accuracy, the first areas that come to mind are data and implementation. Intuitively as more data is introduced into training, the more accurate the results become or at the very least, the more information is provided on how accurate the strategy really is. If the increase in accuracy plateaus, the next step would be to modify the techniques, or the topologies used. As with all machine learning techniques, especially with so many variables and parameters affecting the performance and accuracy, an optimized setting cannot be guaranteed, but with small and gradual tweaks here and there, the bar will be raised.

Generalization is also significant, and can concern within an environment profile or among them. the metrics of generalization within an environment would mostly be a result of human inconsistency and erratic behaviour, as well as there being numerous forms that humans can exist in that environment. Wheelchairs, scooters, bicycles, pedestrians, trucks, farm vehicles, skateboards, children, and many other forms of presence can affect

the generalization ability of any prediction algorithm as each have a different behaviour and it is expected of the autonomous vehicle to be able to navigate any environment as safely as possible. Among environments simply concerns with how many driving scenarios the strategy encompasses.

To summarize:

1. Accuracy enhancements
    - Additional data incorporation
    - Systemic tweaks:
        - Training and network parameters
        - Network topologies
        - CVPM replacement
2. Generalization improvements
    - Inter-environmental generalization
    - Intra-environmental generalization

I would like to conclude this thesis with two points that one of my favourite science authors, Steven Pinker made in a debate about human progress. He recounts ten areas where human life is now better than before and those areas are: life itself, health, prosperity, peace, safety, freedom, knowledge, human rights, gender equality, and intelligence. As surprising as it may sound, autonomous driving can and will affect more than half of these areas for the better. He also reminds the audience that "journalists report plane crashes and not the planes that take off." The numbers speak for themselves. Considering the innumerable efforts to achieve fully autonomous driving and so many human lives at risk, any contribution to the art can be just another baby step toward a breakthrough. Given the matured state of controllers, it is apparent that effort in perception is a more productive way to mitigate the hindrance, halting the progress of autonomous driving. Ultimately, both domains are hoped to have a reinforcing effect on each other, pushing the art forward and bringing autonomous driving and its benefits closer, baby step by baby step.

# References


[1] *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. Feb 2015.

[2] *2015 Motor Vehicle Crashes: Overview*. NHTSAs National Center for Statistics and Analysis, 2016.

[3] D. Watzenig and M. Horn, *Automated Driving Safer and More Efficient Future Driving*. Springer International Publishing, 2018.

[4] Q. Gong, Y. Li, and Z.-R. Peng, "Optimal power management of plug-in hybrid electric vehicles with trip modeling," *Volume 16: Transportation Systems*, 2007.

[5] Y. He, *Vehicle-infrastructure integration enabled plug-in hybrid electric vehicles for energy management*. PhD thesis.

[6] F. A. Bender, M. Kaszynski, and O. Sawodny, "Drive cycle prediction and energy management optimization for hybrid hydraulic vehicles," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 8, p. 35813592, 2013.

[7] H. Chen, L. Guo, H. Ding, Y. Li, and B. Gao, "Real-time predictive cruise control for eco-driving taking into account traffic constraints," *IEEE Transactions on Intelligent Transportation Systems*, p. 111, 2018.

[8] F. Zhang, J. Xi, and R. Langari, "Real-time energy management strategy based on velocity forecasts using v2v and v2i communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, p. 416430, 2017.

[9] B. Yao, C. Chen, Q. Cao, L. Jin, M. Zhang, H. Zhu, and B. Yu, "Short-term traffic speed prediction for an urban corridor," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 2, p. 154169, 2016.

[10] K. Yeon, K. Min, J. Shin, M. Sunwoo, and M. Han, "Ego-vehicle speed prediction using a long short-term memory based recurrent neural network," *International Journal of Automotive Technology*, vol. 20, no. 4, p. 713722, 2019.

[11] E. Thorsell, *Vehicle speed-profile prediction without spatial information.* PhD thesis, 2013.

[12] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016.

[13] 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, p. 187197, 2015.

[14] D. Moser, H. Waschl, R. Schmied, H. Efendic, and L. D. Re, "Short term prediction of a vehicles velocity trajectory using its," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 8, no. 2, p. 364370, 2015.

[15] J. Roos, G. Gavin, and S. Bonnevay, "A dynamic bayesian network approach to forecast short-term urban rail passenger flows with incomplete data," *Transportation Research Procedia*, vol. 26, p. 5361, 2017.

[16] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[17] W. Wang, J. Xi, and D. Zhao, "Driving style analysis using primitive driving patterns with bayesian nonparametric approaches," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, p. 29862998, 2019.

[18] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, p. 6979, 2015.

[19] C. Sun, X. Hu, S. J. Moura, and F. Sun, "Velocity predictors for predictive energy management in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, p. 11971204, 2015.

[20] B. Sakhdari and N. L. Azad, "Adaptive tube-based nonlinear mpc for economic autonomous cruise control of plug-in hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, p. 1139011401, 2018.

[21] M. Vajedi and N. L. Azad, "Ecological adaptive cruise controller for plug-in hybrid electric vehicles using nonlinear model predictive control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, p. 113122, Jan 2016.

[22] Y. L. Murphey, Z. Chen, L. Kiliaris, J. Park, M. Kuang, A. Masrur, and A. Phillips, "Neural learning of driving environment prediction for vehicle power management," *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008.

[23] A. Sarkar, K. Czarnecki, M. Angus, C. Li, and S. Waslander, "Trajectory prediction of traffic agents at urban intersections through learned interactions," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[24] S. Wang, X. Zhang, F. Li, P. S. Yu, and Z. Huang, "Efficient traffic estimation with multi-sourced data by parallel coupled hidden markov model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, p. 30103023, 2019.

[25] M. Zamani Abnili and N. L. Azad, "Short term predictions of preceding vehicle speeds for connected and automated vehicles," *Controls, Dynamic Systems and Robotics (CDSR)*, 2019.

[26] M. Zamani Abnili and N. L. Azad, "A new data-driven approach for on-line traffic participant behaviour prediction at intersections for automated driving," *Canadian Society for Mechanical Engineering International Congress*, 2020.

[27] M. Zamani Abnili and N. L. Azad, "Short-term traffic participants behavior prediction using a novel hybrid method in highway merging for automated vehicles," *(Under Review)*.

[28] S. Tajeddin, S. Ekhtiari, M. Faieghi, and N. L. Azad, "Ecological adaptive cruise control with optimal lane selection in connected vehicle environments," *IEEE Transactions on Intelligent Transportation Systems*, p. 112, 2019.

[29] M. Bayes and M. Price, "An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s.," *Philosophical Transactions (1683-1775)*, vol. 53, pp. 370–418, 1763.

[30] Z. Ghahramani, "Learning dynamic bayesian networks," in *International School on Neural Networks, Initiated by IIASS and EMFCSC*, pp. 168–197, Springer, 1997.

[31] "Us highway 101 dataset, fhwa-hrt-07-030." url=https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm.

[32] J. Pearl, *Bayesian networks*. Computer Science Dept., University of California, 1998.

[33] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, "Traffic speed prediction and congestion source exploration: A deep learning method," *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016.

[34] L. Li, X. Qu, J. Zhang, Y. Wang, and B. Ran, "Traffic speed prediction for intelligent transportation system based on a deep feature fusion model," *Journal of Intelligent Transportation Systems*, p. 112, 2019.

[35] B. Sakhdari and N. L. Azad, "A distributed reference governor approach to ecological cooperative adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, p. 14961507, 2018.

[36] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[37] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, p. 234242, 1992.

[38] R. Raul, *Neural networks: a systematic introduction*. Springer, 1996.

[39] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35, 1960.

[40] R. Berg, "Estimation and prediction for maneuvering target trajectories," *IEEE Transactions on Automatic Control*, vol. 28, no. 3, p. 294304, 1983.

[41] Y. Xie, Y. Zhang, and Z. Ye, "Short-term traffic volume forecasting using kalman filter with discrete wavelet decomposition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 22, no. 5, p. 326334, 2007.

[42] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle," *2007 American Control Conference*, 2007.

[43] Z. Zaidi and B. Mark, "Real-time mobility tracking algorithms for cellular networks based on kalman filtering," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, p. 195208, 2005.

# APPENDICES

# Appendix A

# Cumulative Standardized Normal Distribution Table

NORMAL CUMULATIVE DISTRIBUTION FUNCTION

| $z$ | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 0.5000 | 0.5040 | 0.5080 | 0.5120 | 0.5160 | 0.5199 | 0.5239 | 0.5279 | 0.5319 | 0.5359 |
| 0.1 | 0.5398 | 0.5438 | 0.5478 | 0.5517 | 0.5557 | 0.5596 | 0.5636 | 0.5675 | 0.5714 | 0.5753 |
| 0.2 | 0.5793 | 0.5832 | 0.5871 | 0.5910 | 0.5948 | 0.5987 | 0.6026 | 0.6064 | 0.6103 | 0.6141 |
| 0.3 | 0.6179 | 0.6217 | 0.6255 | 0.6293 | 0.6331 | 0.6368 | 0.6406 | 0.6443 | 0.6480 | 0.6517 |
| 0.4 | 0.6554 | 0.6591 | 0.6628 | 0.6664 | 0.6700 | 0.6736 | 0.6772 | 0.6808 | 0.6844 | 0.6879 |
| 0.5 | 0.6915 | 0.6950 | 0.6985 | 0.7019 | 0.7054 | 0.7088 | 0.7123 | 0.7157 | 0.7190 | 0.7224 |
| 0.6 | 0.7257 | 0.7291 | 0.7324 | 0.7357 | 0.7389 | 0.7422 | 0.7454 | 0.7486 | 0.7517 | 0.7549 |
| 0.7 | 0.7580 | 0.7611 | 0.7642 | 0.7673 | 0.7703 | 0.7734 | 0.7764 | 0.7794 | 0.7823 | 0.7852 |
| 0.8 | 0.7881 | 0.7910 | 0.7939 | 0.7967 | 0.7995 | 0.8023 | 0.8051 | 0.8078 | 0.8106 | 0.8133 |
| 0.9 | 0.8159 | 0.8186 | 0.8212 | 0.8238 | 0.8264 | 0.8289 | 0.8315 | 0.8340 | 0.8365 | 0.8389 |
| 1.0 | 0.8413 | 0.8438 | 0.8461 | 0.8485 | 0.8508 | 0.8531 | 0.8554 | 0.8577 | 0.8599 | 0.8621 |
| 1.1 | 0.8643 | 0.8665 | 0.8686 | 0.8708 | 0.8729 | 0.8749 | 0.8770 | 0.8790 | 0.8810 | 0.8830 |
| 1.2 | 0.8849 | 0.8869 | 0.8888 | 0.8907 | 0.8925 | 0.8944 | 0.8962 | 0.8980 | 0.8997 | 0.9015 |
| 1.3 | 0.9032 | 0.9049 | 0.9066 | 0.9082 | 0.9099 | 0.9115 | 0.9131 | 0.9147 | 0.9162 | 0.9177 |
| 1.4 | 0.9192 | 0.9207 | 0.9222 | 0.9236 | 0.9251 | 0.9265 | 0.9279 | 0.9292 | 0.9306 | 0.9319 |
| 1.5 | 0.9332 | 0.9345 | 0.9357 | 0.9370 | 0.9382 | 0.9394 | 0.9406 | 0.9418 | 0.9429 | 0.9441 |
| 1.6 | 0.9452 | 0.9463 | 0.9474 | 0.9484 | 0.9495 | 0.9505 | 0.9515 | 0.9525 | 0.9535 | 0.9545 |
| 1.7 | 0.9554 | 0.9564 | 0.9573 | 0.9582 | 0.9591 | 0.9599 | 0.9608 | 0.9616 | 0.9625 | 0.9633 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.8 | 0.9641 | 0.9649 | 0.9656 | 0.9664 | 0.9671 | 0.9678 | 0.9686 | 0.9693 | 0.9699 | 0.9706 |
| 1.9 | 0.9713 | 0.9719 | 0.9726 | 0.9732 | 0.9738 | 0.9744 | 0.9750 | 0.9756 | 0.9761 | 0.9767 |
| 2.0 | 0.9772 | 0.9778 | 0.9783 | 0.9788 | 0.9793 | 0.9798 | 0.9803 | 0.9808 | 0.9812 | 0.9817 |
| 2.1 | 0.9821 | 0.9826 | 0.9830 | 0.9834 | 0.9838 | 0.9842 | 0.9846 | 0.9850 | 0.9854 | 0.9857 |
| 2.2 | 0.9861 | 0.9864 | 0.9868 | 0.9871 | 0.9875 | 0.9878 | 0.9881 | 0.9884 | 0.9887 | 0.9890 |
| 2.3 | 0.9893 | 0.9896 | 0.9898 | 0.9901 | 0.9904 | 0.9906 | 0.9909 | 0.9911 | 0.9913 | 0.9916 |
| 2.4 | 0.9918 | 0.9920 | 0.9922 | 0.9925 | 0.9927 | 0.9929 | 0.9931 | 0.9932 | 0.9934 | 0.9936 |
| 2.5 | 0.9938 | 0.9940 | 0.9941 | 0.9943 | 0.9945 | 0.9946 | 0.9948 | 0.9949 | 0.9951 | 0.9952 |
| 2.6 | 0.9953 | 0.9955 | 0.9956 | 0.9957 | 0.9959 | 0.9960 | 0.9961 | 0.9962 | 0.9963 | 0.9964 |
| 2.7 | 0.9965 | 0.9966 | 0.9967 | 0.9968 | 0.9969 | 0.9970 | 0.9971 | 0.9972 | 0.9973 | 0.9974 |
| 2.8 | 0.9974 | 0.9975 | 0.9976 | 0.9977 | 0.9977 | 0.9978 | 0.9979 | 0.9979 | 0.9980 | 0.9981 |
| 2.9 | 0.9981 | 0.9982 | 0.9982 | 0.9983 | 0.9984 | 0.9984 | 0.9985 | 0.9985 | 0.9986 | 0.9986 |
| 3.0 | 0.9987 | 0.9987 | 0.9987 | 0.9988 | 0.9988 | 0.9989 | 0.9989 | 0.9989 | 0.9990 | 0.9990 |
| 3.1 | 0.9990 | 0.9991 | 0.9991 | 0.9991 | 0.9992 | 0.9992 | 0.9992 | 0.9992 | 0.9993 | 0.9993 |
| 3.2 | 0.9993 | 0.9993 | 0.9994 | 0.9994 | 0.9994 | 0.9994 | 0.9994 | 0.9995 | 0.9995 | 0.9995 |
| 3.3 | 0.9995 | 0.9995 | 0.9995 | 0.9996 | 0.9996 | 0.9996 | 0.9996 | 0.9996 | 0.9996 | 0.9997 |
| 3.4 | 0.9997 | 0.9997 | 0.9997 | 0.9997 | 0.9997 | 0.9997 | 0.9997 | 0.9997 | 0.9997 | 0.9998 |
| 3.5 | 0.9998 | 0.9998 | 0.9998 | 0.9998 | 0.9998 | 0.9998 | 0.9998 | 0.9998 | 0.9998 | 0.9998 |
| 3.6 | 0.9998 | 0.9998 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 |
| 3.7 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 |
| 3.8 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 | 0.9999 |
| 3.9 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |

# Appendix B

# Back Propagation Algorithm

Back propagation learning is a method of learning in which the errors are propagated from the outermost layer inwards and the weights of the network are adjusted based on the difference between the desired outputs and the network outputs. To put the method into perspective, suppose the desired value for neuron $j$ is known. $w_{ij}$ is the input weight going from neuron $i$ to neuron $j$ (needless to say, $i$ is in one layer before $j$). $x_j$ is the input to neuron $j$, $y_j$ is the corresponding output and $d_j$ is the desired output. based on (2.18), $x_j$ can be computed as:

$$x_j = \sum_i y_i w_{ij} \tag{B.1}$$

and assuming the logistic sigmoid activation function, the output is computed as:

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{B.2}$$

For the specific training case ($c$) the desired output is known, therefore the global error can be defined as the squared difference of the output and the desired value:

$$E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2 \tag{B.3}$$

To start the BP-Algorithm, the network is utilized to compute the outputs for a training case going forward, initialized by a random set of weights. The input vector presented to the network from the training case is associated with a desired output. After the network has computed the outputs for every input presented, the global error ($E$) can be computed. The error computed then can be utilized to update the weights in such a way that the errors

are minimized. Mathematically, the minima of error with respect to the weights can be found by finding where the derivative of error with respect to weights becomes zero. The weights can then be updated proportional to the negative of this derivation, mathematically represented as:

$$\Delta w_{ij,c} \propto -\frac{\partial E_c}{\partial w_{ij,c}} \tag{B.4}$$

the only task that remains is to find the right hand side in (B.4). Using chain rule we can write:

$$\frac{\partial E_c}{\partial w_{ij,c}} = \frac{\partial E_c}{\partial x_{j,c}} . \frac{\partial x_{j,c}}{\partial w_{ij,c}} \tag{B.5}$$

where $x_{j,c}$ denotes the input to neuron $j$ and can be computed as:

$$x_{j,c} = \sum_k w_{kj} y_{k,c} = \sum_{k \neq i} w_{kj} y_{k,c} + w_{ij} y_{i,c} \tag{B.6}$$

substituting (B.6) into (B.5) one partial derivative can be calculated as:

$$\frac{\partial x_{j,c}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (\sum_{k \neq i} w_{kj} y_{k,c} + w_{ij} y_{i,c}) = y_{i,c} \tag{B.7}$$

for the other partial derivative we can define "error signal ($\delta$):

$$\delta_{j,c} = -\frac{\partial E_c}{\partial x_{j,c}} \tag{B.8}$$

by adding "learning rate ($\eta$) to the mix to adjust the rate of which the weights are updated to travel along the gradient, the weight updates can be represented as:

$$\Delta w_{ij,c} = \eta \ . \ \delta_{j,c} \ . \ y_{i,c} \tag{B.9}$$

To evaluate the error signal ($\delta$), chain rule can be applied a second time.

$$\frac{\partial E_c}{\partial x_{j,c}} = \frac{\partial E_c}{\partial y_{j,c}} \frac{\partial y_{j,c}}{\partial x_{j,c}} \tag{B.10}$$

given the activation function $\varphi$ the relationship between $y_{j,c}$ and $x_{j,c}$ can be expressed as $y_{j,c} = \varphi(x_{j,c})$, and therefore:

$$\frac{\partial y_{j,c}}{\partial x_{j,c}} = \varphi'(x_{j,c})$$

for the first derivative in (B.10) two cases need to be considered.

1. $j$ is an output neuron (i.e. the training case explicitly defines the desired value for every input to the neuron): In this case, the output error is known as a function of the network output ($y_{j,c}$). Therefore:

$$\frac{\partial E_c}{\partial y_{j,c}} = y_{j,c} - d_{j,c} \tag{B.11}$$

and the error signal follows:

$$\delta_{j,c} = (y_{j,c} - d_{j,c}) \; . \; \varphi'(x_{j,c}) \tag{B.12}$$

2. $j$ is a hidden neuron. In this case, the error is not explicitly known. However, after applying the first case to the output layer ($l$), we know:

$$E_c = \frac{1}{2} \sum_l e_{l,c}^2 \tag{B.13}$$

where $e_{l,c}$ represents:

$$e_{l,c} = d_{l,c} - y_{l,c} \tag{B.14}$$

going back to the derivative, we can write:

$$\frac{E_c}{y_{j,c}} = \sum_l e_{l,c} . \frac{\partial e_{l,c}}{\partial y_{j,c}} \tag{B.15}$$

by applying the chain rule once more, the above equation can be written as:

$$\frac{\partial E_c}{\partial y_{j,c}} = \sum_l e_{l,c} . \frac{\partial e_{l,c}}{\partial x_{l,c}} . \frac{\partial x_{l,c}}{\partial y_{j,c}} \tag{B.16}$$

It is known that:

$$\begin{aligned} e_{l,c} &= d_{l,c} - y_{l,c} \\ &= d_{l,c} - \varphi(x_{l,c}) \end{aligned} \tag{B.17}$$

hence,

$$\frac{\partial e_{l,c}}{\partial x_{l,c}} = -\varphi'(x_{l,c}) \tag{B.18}$$

From (B.1) we know that we can write $x_{l,c}$ as:

$$x_{l,c} = \sum_j w_{jl,c} . y_{j,c} \tag{B.19}$$

differentiating the equation with respect to $y_{j,c}$:

$$\frac{\partial x_{l,c}}{\partial y_{j,c}} = w_{jl,c} \tag{B.20}$$

$$\begin{aligned} \frac{\partial E_c}{y_{j,c}} &= -\sum_l e_{l,c}.\varphi(x_{l,c}).w_{jl,c} \\ &= -\sum_l \delta_{j,c}.w_{jl,c} \end{aligned} \tag{B.21}$$

Finally, by rearranging the terms we can find $\delta_{j,c}$ as:

$$\delta_{j,c} = \varphi'(x_{j,c}).\sum_l \delta_{l,c}.w_{jl,c} \tag{B.22}$$

Having found $\delta_{j,c}$ for both cases, the equation (B.9) can be used to update the weights in such a way to decrease error over iterations. As mentioned before, the activation function used in this study is sigmoid function. Therefore for this case, the solution can be particularized to the specific function as follows:

$$y_{j,c} = \varphi(x_{j,c}) = \frac{1}{1+e^{-x_{j,c}+\theta}} \tag{B.23}$$

To calculate $\varphi'(x_{j,c})$, we can write:

$$\begin{aligned} \varphi'(x_{j,c}) &= \frac{\partial}{\partial x_{j,c}}\left(\frac{1}{1+e^{-x_{j,c}+\theta}}\right) \\ &= \left(\frac{-1}{(1+e^{-x_{j,c}+\theta})^2}\right).\frac{\partial}{\partial x_{j,c}}(1+e^{-x_{j,c}+\theta}) \\ &= \left(\frac{-1}{(1+e^{-x_{j,c}+\theta})^2}\right)e^{-x_{j,c}+\theta}.\frac{\partial}{\partial x_{j,c}}(x_{j,c}+\theta) \\ &= \left(\frac{1}{1+e^{-x_{j,c}+\theta}}\right).\left(\frac{e^{-x_{j,c}+\theta}}{1+e^{-x_{j,c}+\theta}}\right) \\ &= \left(\frac{1}{1+e^{-x_{j,c}+\theta}}\right).\left(\frac{1+e^{-x_{j,c}+\theta}}{1+e^{-x_{j,c}+\theta}} - \frac{1}{1+e^{-x_{j,c}+\theta}}\right) \\ &= \left(\frac{1}{1+e^{-x_{j,c}+\theta}}\right).\left(1 - \frac{1}{1+e^{-x_{j,c}+\theta}}\right) \end{aligned}$$

By substituting $y_{j,c}$ into above equation we will get:

$$\varphi'(x_{j,c}) = y_{j,c}.(1-y_{j,c}) \tag{B.24}$$

going back to equations (B.12) and (B.22), $\delta_{j,c}$ can be computed as follows:

$$\begin{cases} \delta_{j,c} = (d_{j,c} - y_{j,c}).y_{j,c}.(1 - y_{j,c}) & \text{for output neurons} \\ \delta_{j,c} = y_{j,c}.(1 - y_{j,c}).\sum_l \delta_{l,c} w_{j,l} & \text{for hidden neurons} \end{cases} \tag{B.25}$$

The training can be iteratively carried out until the global error falls below a certain threshold.