Traffic Light Status Detection Using Movement Patterns of Vehicles

by

Joseph Campbell

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2016 by the
Graduate Supervisory Committee:

Georgios Fainekos, Chair
Heni Ben Amor
Panagiotis Artemiadis

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

Traditional methods for detecting the status of traffic lights used in autonomous vehicles may be susceptible to errors, which is troublesome in a safety-critical environment. In the case of vision-based recognition methods, failures may arise due to disturbances in the environment such as occluded views or poor lighting conditions. Some methods also depend on high-precision meta-data which is not always available. This thesis proposes a complementary detection approach based on an entirely new source of information: the movement patterns of other nearby vehicles. This approach is robust to traditional sources of error, and may serve as a viable supplemental detection method. Several different classification models are presented for inferring traffic light status based on these patterns. Their performance is evaluated over real-world and simulation data sets, resulting in up to 97% accuracy in each set.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Georgios Fainekos, for the incredible research opportunities and mentorship he has provided over the past several years. I would like to thank Dr. Heni Ben Amor, Dr. Panagiotis Artemiadis, Dr. Theodore Pavlic, Dr. Umit Ogras, and Dr. Yu Zhang for the invaluable advice and guidance they have given me. I would like to thank Dr. Marcelo Ang for the amazing research experience at the National University of Singapore that made this work possible, as well as everyone at SMART who graciously provided assistance and made me feel so welcome, in particular: Dr. James Fu, Dr. Baoxing Qin, and Scott Pendleton. I would to thank my co-authors and labmates at the Cyber Physical Systems Lab for the wonderful discussions and collaborations: Kangjin Kim, Bardh Hoxha, Erkan Tuncali, Ujjwal Gupta, Adel Dokhanchi, Rahul Srinivasa, Shakiba Yaghoubi, Wei Wei, and Dylan Lusi.

This material is based upon work supported by the National Science Foundation under EAPSI Fellowship Grant No. 1515589 and CPS 1446730. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Autonomous vehicles are a promising technology, potentially yielding many societal benefits such as fewer traffic-related fatalities, reduced pollution and energy consumption, and greater mobility to those incapable of operating a standard automobile. However, these are complex systems that must be capable of interacting with human-operated automobiles, pedestrians, and infrastructure, not to mention other intelligent systems. Unsurprisingly, there are significant challenges that must be overcome before autonomous vehicles can be considered safe for widespread introduction into present-day road networks.

One such challenge is the ability to safely navigate complex environments such as intersections while maintaining compliance with local traffic regulations. Vehicles and pedestrians with varying directions of travel cross paths while being guided by traffic lights that are optimized for identification by human drivers. This issue has been partially addressed with the introduction of intelligent traffic light systems which actively communicate their signal to nearby vehicles through Vehicular Ad Hoc Networks [23, 11]. Nonetheless, as observed by [21], this remains an open problem as such systems have thus far been limited to small-scale academic experiments and a timely integration into current road networks seems unlikely.

It is for this reason that recent work has focused on the real-time identification of traffic light signals via vision-based systems [9]. This approach can work well but is subject to errors that can lead to the misidentification of traffic light signals. These errors can arise due to poor lighting conditions which interfere with the camera sensor or an obstructed view resulting from a dirty lens or another vehicle, as demonstrated

Figure 1.1: Example of traffic light occlusion by other vehicles. The light is obscured in the top image but becomes visible in the bottom image as the traffic starts moving.

in Fig. 1.1. This can potentially lead to disastrous results – an autonomous vehicle erroneously passing through an intersection could find itself in a situation in which it is unable to avoid a collision.

1.1 Contributions

This thesis proposes a complementary traffic light identification system that uses an alternative source of information: the behavior of other nearby vehicles based on positional data. Conceptually, the system infers the status of a traffic light from the movements of other vehicles around or in the corresponding intersection. The advantage of such a system is not that it has fewer failure-inducing cases than a vision-based system, but rather that they are *different* failure cases. Ideally, this system will be paired with a vision-based one such that they complement each other and reduce the total points of failure.

Consider the following scenario: a traffic light is non-functional due to an extenuating circumstance. As is typical on US roads, a law enforcement officer is directing traffic through the intersection. A typical vision-based recognition system is of no help in this scenario, however, by observing when other vehicles start to pass through the intersection and from which direction, the proposed system can infer which traffic the officer is allowing to pass through the intersection.

Similarly, consider a situation in which a vehicle stops at a red light behind a larger vehicle that is occluding the traffic light. Suppose the preceding vehicle suffers a mechanical failure and is blocking traffic; the traffic light cycles through its phases and surrounding traffic bypasses the offending vehicle in other lanes. Once again, a vision-based system would not be of assistance in this scenario, however, the proposed system may indicate that the traffic light is green and therefore alternative action should be taken.

The contributions of this thesis are as follows: we formally define the problem, present a system for predicting the state of a traffic light based on the spatial movement of nearby vehicles, and evaluate its effectiveness in simulated and real-world conditions.

1.2 Summary of Publications

A subset of the most salient results from this thesis has been submitted in the following publication and is currently under review.

- **J. Campbell**, H.B. Amor, M.H. Ang Jr. and G. Fainekos, **Traffic Light Status Detection Using Movement Patterns of Vehicles** under review in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016 [4]

The following publications are not necessarily directly related to this thesis topic, however they represent a significant amount of time and effort. Therefore, this section will briefly describe the publications that the author has contributed to during the course of this degree.

- K. Kim, **J. Campbell**, W. Duong, Y. Zhang and G. Fainekos, **DisCoF$^+$: Asynchronous DisCoF with Flexible Decoupling for Cooperative Pathfinding** in 2015 IEEE International Conference on Automation Science and Engineering (CASE), 2015 [19]

  This work expands on the previously introduced distributed cooperative multi-robot path planning algorithm DisCoF [36]. In DisCoF, robots initially plan independently and dynamically *couple* together into groups when there is risk of a path conflict, which occurs when more than one robot attempts to occupy the same position at the same time. When coupled, robots plan together so as to avoid these conflicts, yet by coupling only when necessary the search space remains small. The extension, DisCoF$^+$, adds the notion of *decoupling* in which robots are no longer grouped together when the risk of a path conflict passes. Additionally, the prior requirement that robots must operate in synchronized time steps has been removed, allowing asynchronous planning. My contribution to this work was an experimental simulation in which multiple iRobot Creates employed the DisCoF$^+$ algorithm to cooperatively plan in an obstacle-filled environment in the Webots [24] simulator.

- U. Gupta, **J. Campbell**, U.Y. Ogras, R. Ayoub, M. Kishinevsky, F. Paterna and S. Gumussoy, **Adaptive Performance Prediction for Integrated GPUs** to appear in 2016 IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2016 [15]

In this paper we introduce an adaptive prediction model based on Recursive Least Squares which is capable of accurately predicting GPU performance in an efficient manner at run-time. This is particularly relevant for power management in embedded devices such as smart phones, as an accurate performance model enables Dynamic Voltage and Frequency Scaling (DVFS) algorithms. My contribution was to assist in the development of the prediction model, perform experiments on a physical platform, and analyze the results.

- **J. Campbell**, C.E. Tuncali, P. Liu, T.P. Pavlic, U. Ozguner and G. Fainekos, **Modeling Concurrency and Reconfiguration in Vehicular Systems: A π-Calculus Approach** to appear in 2016 IEEE International Conference on Automation Science and Engineering (CASE), 2016 [5]

  J. Campbell, C.E. Tuncali, T.P. Pavlic and G. Fainekos, **Toward Modeling Concurrency and Reconfiguration in Vehicular Systems** in 9th Interaction and Concurrency Experience, Satellite Workshop of DisCoTec 2016, 2016 [6]

  This work introduces a hierarchical modeling framework for cooperation among autonomous vehicles. Supervisory communication and control is handled by a high-level layer via π-calculus expressions, and low-level dynamics and continuous control are defined by hybrid automata. My contribution was the initial development of the framework, definition of the π-calculus expressions and hybrid automata for a vehicle platoon case study, and experimental analysis of the framework.

Chapter 2

RELATED WORK

Vision-based traffic light detection systems have been widely analyzed in previous works. The majority of these works have focused purely on image recognition [25, 34, 22]. Of particular interest, however, are those that seek to minimize the risk posed by errors inherent to vision-based detection systems. In [9], the authors propose using a detailed map of traffic lights to act as prior knowledge so that the detection system knows when it should be able to see traffic lights. If traffic lights are not detected at an expected position, the autonomous vehicle can take preventative action such as slowing down under the assumption that the light is red or yellow. If the light is actually red or yellow, then this course of action is the correct one. If the light is actually green then the vehicle slowing is an annoyance at best, and results in a collision with human-operated vehicles due to unpredictability at worst. If the system has a poor detection rate this could start to degrade the flow of traffic along a road as vehicles slow down for a green light. In addition, this approach requires detailed prior map knowledge of the traffic lights.

Similarly, in [21] the authors acknowledge the difficulties in building a purely vision-based traffic light detection system and so augment theirs with prior map knowledge as well as temporal information. While yielding good results, the system still fails to identify traffic light signals in certain cases. Indeed, the authors indicate that a possible approach for improvement would be to introduce 3-dimensional LIDAR data into the mix in order to improve recognition of the traffic lights themselves.

In [26], the authors use the movement patterns of pedestrians to apply semantic labels to the environment. They infer the location of pedestrian crossings, sidewalks,

and building entrances and exits based on the activity patterns of pedestrians. This is similar in spirit, if not in execution, to the labeling of traffic lights based on vehicle movement patterns introduced in this thesis.

Chapter 3

PROBLEM FORMULATION

We can formalize the problem from a probabilistic perspective as follows: let Z be a discrete random variable which represents the state of a traffic light with respect to a target vehicle. The specific value of Z is denoted by z, and in this work can take the value of either *green* or *red*. The goal is to then determine the probability that a traffic light is either *green* or *red* with respect to our target vehicle at a specific point in time t: $p(Z_t = z_t)$. To simplify the notation, from this point on we will refer to this probability as simply $p(z_t)$.

Clearly we cannot determine an accurate probability for $p(z_t)$ without additional

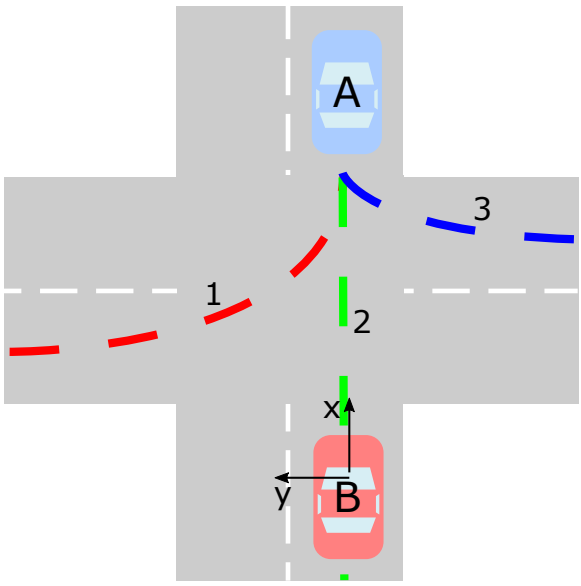

Figure 3.1: A scenario in which vehicle A's current position is ambiguous, as there are multiple paths it could have taken which could be used to infer different traffic light states.

information. Therefore, we would like to consider observations of nearby vehicles when determining this probability. The simplest approach is to consider the spatial position of every nearby vehicle independently at each point in time. We define the state $\mathbf{g}$ of vehicle n at time t as a vector:

$$\mathbf{g}_{n,t} = \begin{bmatrix} x_{n,t}, y_{n,t} \end{bmatrix} \tag{3.1}$$

If we place the target vehicle at the origin of a Cartesian coordinate plane with the positive x-axis extending towards the front of the vehicle and the positive y-axis extending towards the left-hand side of the vehicle, then x is the distance along the x-axis from the target vehicle to the observed vehicle n. Similarly, y is the distance along the y-axis to vehicle n. This yields a conditional probability of the following form, where N is the total number of observable vehicles at time t.

$$p(z_t|\mathbf{g}_{1,t}, \mathbf{g}_{2,t}, ..., \mathbf{g}_{N,t}) = p(z_t|\mathbf{g}_{1:N,t}) \tag{3.2}$$

However, this approach has a significant drawback which is visualized in Fig. 3.1. If vehicle A is an observable vehicle at time t, it may have taken several different paths to arrive at this position: path 1, 2, or 3. Each of these paths could result in a different traffic light state z_t. For example, if vehicle B is our target vehicle and we are observing A, z_t could hold the value of *green* if vehicle A followed path 2, *red* for path 1, and either *green* or *red* for path 3 (depending on local traffic regulations for right-on-red turns). This leads to an ambiguous situation, in which the state of vehicle A at this point in time does not necessarily help us determine z_t.

We can alleviate this problem if we consider a temporal trace of the position. We could alter the vehicle state to include information on the position over time in the form of velocity.

$$\mathbf{g}_{n,t} = \left[x_{n,t}, y_{n,t}, \dot{x}_{n,t}, \dot{y}_{n,t} \right] \tag{3.3}$$

This is susceptible to the same ambiguity problem, however. In the example from Fig. 3.1, if path 2 resulted from A accelerating through a light which recently turned *green*, then the velocity at time t could be roughly the same for all paths. The same holds true when acceleration is considered.

$$\mathbf{g}_{n,t} = \left[x_{n,t}, y_{n,t}, \dot{x}_{n,t}, \dot{y}_{n,t}, \ddot{x}_{n,t}, \ddot{y}_{n,t} \right] \tag{3.4}$$

A more effective approach is to consider the state of vehicle n not just for a single time step t, but rather over a time window, i.e., $t-1$, $t-2$, and so on. If we consider a window size of T time steps in the past, then we can represent the state of vehicle n as a time series s at time t.

$$\mathbf{s}_{n,t} = \mathbf{g}_{n,t}, \mathbf{g}_{n,t-1}, ..., \mathbf{g}_{n,t-T_n} \tag{3.5}$$

$$p(z_t|\mathbf{s}_{1,t}, \mathbf{s}_{2,t}, ..., \mathbf{s}_{N,t}) = p(z_t|\mathbf{s}_{1:N,t}) \tag{3.6}$$

If observations are ideal, then the entire path for vehicle A is now taken into account and there is no more ambiguity. In practice, this may not be the case and the effective window size T_n may vary from vehicle to vehicle. For example, A may only enter the sensor range of our target vehicle once it reaches the position depicted in Fig. 3.1. We now provide a formal problem statement and define our assumptions.

Problem: Given a set of observations $\mathcal{L}$ of nearby vehicles, determine $p(z_t|\mathcal{L})$.

1. $\mathcal{L}$ is either a set of independent vehicle states $\mathbf{g}$, or a set of independent series of states $\mathbf{s}$.

2. Each series $\mathbf{s}$ may contain a variable number of states, however, they must correspond to sequential time points.

3. At least one vehicle must be observed for at least one time step.

In practice, Assumption 2 is not strong as this is implicitly satisfied by a Bayesian tracking algorithm in this work.

Chapter 4

METHODOLOGY

The problem we have defined is in general known as a classification problem: given an input, classify it by applying a discrete label. In this case, the input to our problem is a set of vehicle observations $\mathcal{L}$ and we wish to label it with a discrete value z_t, either *green* or *red*. This work tackles the classification problem via a method known as *supervised learning* [3]. In supervised learning, an existing data set of labeled inputs is used to infer the label of other inputs. For example, we could collect the labeled data set – known as a *training* set – by observing nearby vehicles at an intersection and applying labels by directly observing the traffic light. We can then use this data set to infer the labels for future observations where the traffic light is not observable. The rest of this chapter will provide an introduction to several classification methods and how they are employed in this work. For simplicity, the examples will assume the input data is the most basic vehicle state as defined in Eq. 3.1.

4.1 Nearest Neighbor

So how can the training data be used to infer labels for other observations? The simplest method for doing so is to directly compare the new observation to the training data and use the most populous label among the closest training inputs [7], as measured by a distance metric. Figure 4.1 shows the three closest neighbors to the new observation A as determined by euclidean distance. Once the neighbors are identified, it is a simple matter of "majority rule"; the label that is used by the most neighbors is applied to the new point. In Fig. 4.1, two of the three closest neighbors are *green*, therefore we say that the new point's label is also *green*. This method is

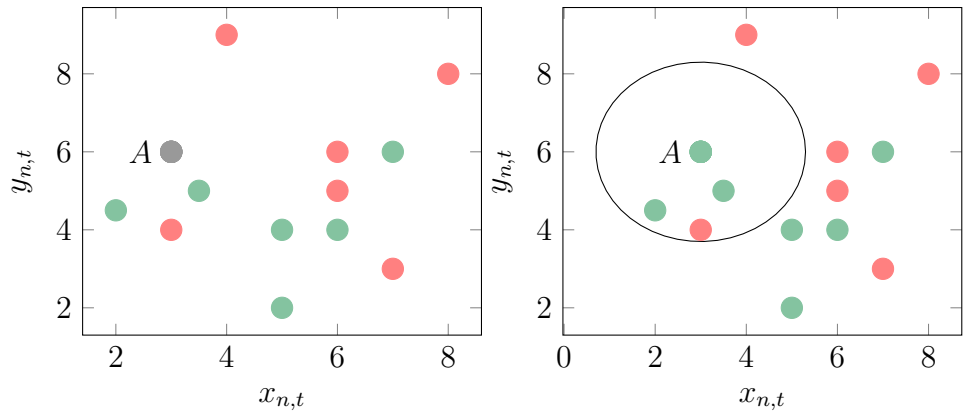

Figure 4.1: Left: We would like to classify the new observation point, A. Right: the three nearest neighbors to A are circled. Since the majority of neighbors are labeled *green*, then so too is A.

known as k-Nearest Neighbors (k-NN), where k is the number of neighbors that is used – typically an odd number so that a majority is guaranteed. One way to view this method is that it implicitly divides the input space up into discrete regions, with each region corresponding to a specific label [3]. New points are given the label that corresponds to the region in which they lie. The borders that separate regions are known as decision boundaries.

The disadvantages of this method are two-fold: k-NN is not probabilistic, meaning that it infers z_t directly given $\mathcal{L}$ and not $p(z_t|\mathcal{L})$; and k-NN is a special type of non-parametric model that must store and operate over all training data instances [3]. The latter point is especially problematic, as we would like a model that can operate in real-time on an autonomous vehicle. This places an upper limit on how many training data instances can be stored and may adversely affect the ability of k-NN to generalize.

4.2 Artificial Neural Networks

Therefore, we turn to parametric models whose computational and storage requirements for classification do not scale with the number of training data instances. Artificial neural networks are parametric mathematical models capable of accurately approximating any continuous function [10]. More specifically, it has been previously shown that ANNs can accurately approximate a Bayesian posterior [27], depending on the network complexity and cost function. This indicates that unlike k-NN, ANNs can efficiently approximate $p(z_t|\mathcal{L})$, making them an ideal classification method for this problem.

4.3 Single-layer Perceptron

Single-layer perceptrons [28] are the simplest type of artificial neural network [16]. The general idea is that we want to construct a function – known as a *discriminant* – which accepts an input and produces a classification label. If there are only two labels, the most basic discriminant is simply a linear combination of the inputs [2].

$$o_{n,t}(\mathbf{g}_{n,t}) = w_0 x_{n,t} + w_1 y_{n,t} = \mathbf{w}^T \mathbf{g}_{n,t} \tag{4.1}$$

If $o_{n,t} > 0$, then the input is given one label; for $o_{n,t} < 0$, the other label. The coefficients w_0 and w_1 are referred to as weights and are considered parameters of the model. These coefficients are determined during a training phase which utilizes the training data set. This model is easily extended to an arbitrary number of labels

$$o_{z,n,t}(\mathbf{g}_{n,t}) = \mathbf{w}_z^T \mathbf{g}_{n,t} \tag{4.2}$$

where the label is determined by the largest value of o

$$\max o_{z,n,t} \tag{4.3}$$

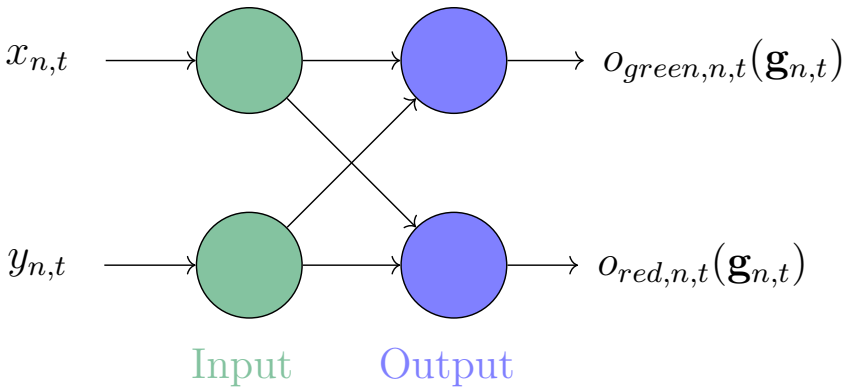

Figure 4.2: Single-layer Perceptron.

This model can be viewed in terms of layered network of nodes as shown in Fig. 4.2. For a single-layer perceptron, there is only one layer which performs computations: the output layer. This is also referred to as a *feedforward* neural network (FFNN) since the network connections are acyclic and directed from input to output.

This model can be further generalized by passing the linear combination of inputs to a nonlinear *activation* function f.

$$o_{z,n,t}(\mathbf{g}_{n,t}) = f(\mathbf{w}_z^T \mathbf{g}_{n,t}) \tag{4.4}$$

It has been shown [27, 2] that for a logistic sigmoid or softmax activation function the outputs accurately approximate a posterior probability. Thus, the single-layer perceptron yields the desired probability

$$p(z_t|\mathbf{g}_{n,t}) = f(\mathbf{w}_z^T \mathbf{g}_{n,t}) \tag{4.5}$$

However, owing to its simplicity, the single-layer perceptron is limited in its classification ability. Though a nonlinear activation function is used, it is monotonic [2]. The result is that the single-layer perceptron is a linear discriminant and can only

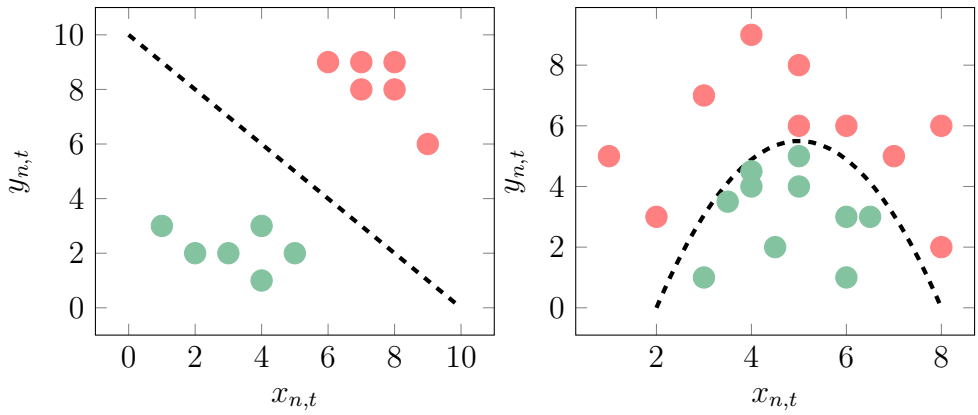

Figure 4.3: Left: a linearly separable data set. Right: a linearly inseparable data set.

generate linear decision boundaries. This makes it relatively ineffective unless the training data is *linearly separable*, i.e., points with different labels can be separated by a straight line as in Fig. 4.3.

4.4 Multilayer Perceptron

A multilayer perceptron [29] is simply a single-layer perceptron with one or more additional layers between the input and output. By taking linear combinations of the inputs more than once in succession, multilayer perceptrons are capable of producing nonlinear decision boundaries. In fact, with just one additional layer (two layers total) a multilayer perceptron can approximate any continuous function [10]. These additional layers are referred to as *hidden* layers, and result in a network such as in Fig. 4.4. Following the notation in [3], the linear combination of inputs for any given node of a multilayer perceptron can be generalized as

$$a_k = \sum_j w_{k,j} a_j \tag{4.6}$$

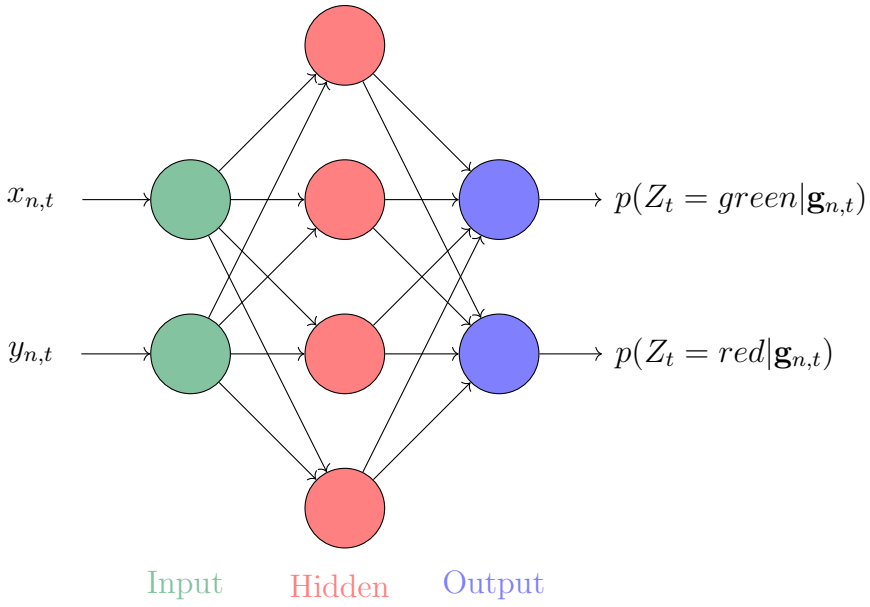

Figure 4.4: Multilayer Perceptron.

which is transformed by a nonlinear activation function f to yield the output value of node k

$$o_k = f(\sum_j w_{k,j} o_j) \tag{4.7}$$

where o_k is the output value of node k with j connecting nodes from the previous layer. By applying this equation recursively, it is possible to find the output values of the network.

It was previously mentioned that the weights of a network are determined during a training phase. More specifically, we use an algorithm that feeds the network inputs from the training data and compares the network's output to the actual output. The network weights are then iteratively adjusted in order to minimize the output error. This is commonly accomplished with an algorithm known as *backpropagation* [30].

Backpropagation consists of two steps: propagate the training inputs forward through the network starting at the input layer to compute the output of every node,

then propagate the error backwards through the network starting at the output layer and use it to update the weights of each node. The first step simply requires applying Eq. 4.6 to each node for every input in the training data. For the second step we must first determine a suitable measure of error for the network. With a softmax activation function used in the output layer of the network, the error can be expressed as the cross entropy of K network outputs and the expected outputs t from a single training instance [3].

$$E = \sum_{k=1}^{K} t_k log(o_k) \tag{4.8}$$

We are interested in the partial derivative of this error with respect to the network weights so that we can evaluate how the weights contribute to the error.

$$\frac{\partial E}{\partial w_{k,j}} = \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{k,j}} = \delta_k o_j \tag{4.9}$$

The node error δ is what we will use to modify the weights of the network. For nodes in the output layer, this is simply the difference in the network outputs and the expected output.

$$\delta_k = o_k - t_k \tag{4.10}$$

In the hidden layers, it is expressed as the differentiated nonlinear activation function multiplied by the weighted errors in the subsequent nodes (remember that we are propagating backward).

$$\delta_j = f'(a_j) \sum_k w_{k,j} \delta_k \tag{4.11}$$

Once δ has been found for each node in the network, it is used to calculate the change in weights.

$$\Delta w_{k,j} = \alpha \delta_k o_j \tag{4.12}$$

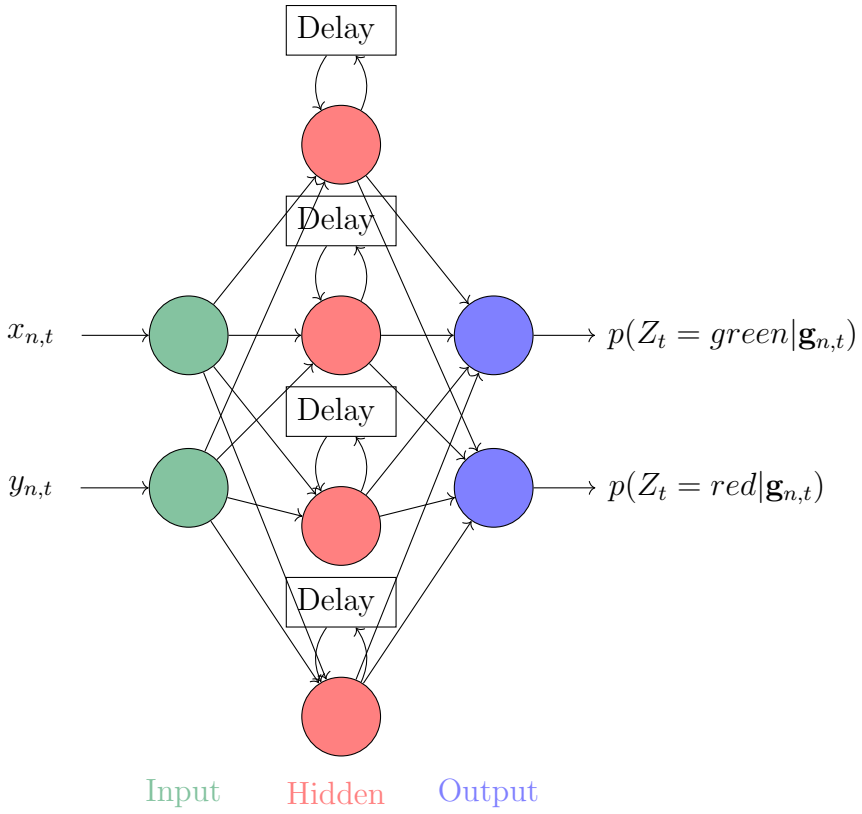

Figure 4.5: Recurrent Neural Network

4.5 Recurrent Neural Networks

The examples until this point have used the independent vehicle states $\mathbf{g}$ defined in Eq. 3.1 as inputs. So how can we classify with the time series of states $\mathbf{s}$ defined in Eq. 3.5? In a feedforward neural network such as a multilayer perceptron, the nodes are not allowed to form cycles; inputs propagate through the network layers from the input layer to the output layer. This is suitable for classification when separate input vectors are treated as independent, however, it is not ideal when we would like to consider some inputs as dependent and use multiple inputs to derive a single output. This is the case when approximating the posterior probability in Eq. (3.6), as it is conditional on a time series of vehicle states $\mathbf{s}_{n,t}$ as defined in Eq. (3.5). This can

be accomplished with recurrent neural networks (RNNs) [8, 18], which are a variant of feedforward networks that are allowed to form cyclical connections among hidden layer nodes as shown in Fig. 4.5. Simply speaking, this allows an RNN to produce an output from a sequence of prior inputs, e.g., the time series in Eq. (3.5), as opposed to a single input.

The self-connections in the hidden layer nodes are associated with a time delay and essentially act as a sort of memory to retain the hidden layer output values for the previous input. Suppose we have the first input of a time series at $t - T_n$. Each hidden layer node will receive the input values along with its previous output value from the self-connection. However, since this is the first input there is no previous output value and so it receives the initial state instead. The final network output at this point is typically discarded, since we are only interested in the output once all inputs in the series have been processed. For the next input at $t - T_n + 1$, the hidden layer nodes receive the new inputs as well as the output values they calculated for $t - T_n$. This process continues until all inputs have been processed, at which point the network output is used.

Recurrent neural networks are still trained with backpropagation, however, an extra step [35] is necessary due to the cyclical connections. A copy of the network is made for each input in the time series and the cyclical connections are replaced with a connection to this copy, essentially forming a multilayer perceptron with many layers. This is known as *unfolding* and allows backpropagation to function just as it would without the cyclical connections.

We use RNNs to estimate $p(z_t|\mathbf{s}_{n,t})$ by generating a single output z_t from a sequence of feature vectors $\mathbf{g}_{n,t}$, which together form the time series $\mathbf{s}_{n,t}$ of state vectors for vehicle n. This is known as *sequence classification* [13, 12]. However, this is not the same as the posterior probability defined in Eq. (3.6) which is conditional on all

observed vehicles, not just one. The feature vector to our network must be a constant size. This rules out simply concatenating the feature vectors of all observed vehicles, since the number of observed vehicles may vary at any given time. Instead, we can take the mean probability of $p(z_t|\mathbf{s}_{n,t})$ for all observed vehicles at time t and use that as an approximation.

$$\hat{p}(z_t|\mathbf{s}_{1:N,t}) \approx \frac{\sum_{n=1}^{N} p(z_t|\mathbf{s}_{n,t})}{N} \tag{4.13}$$

4.6 Bidirectional Long Short Term Memory Networks

A variant of the RNN known as the Bidirectional Long Short Term Memory (BLSTM) network was shown to be exceptionally well-suited for sequence classification [14]. Standard RNNs suffer from a problem known as the *vanishing gradient* [1], in which the hidden layer node weights for previous inputs converge to zero over time, thus preventing an RNN from effectively learning from inputs that span a long time period. The Long Short Term Memory (LSTM) [17] network was designed to mitigate this problem by introducing the concept of LSTM nodes that are more effective at retaining previous values. Long Short Term Memory nodes are themselves a composite of connected nodes, and they replace (some) standard nodes in a recurrent neural network.

The bidirectional aspect of a BLSTM network is a concept taken from Bidirectional Recurrent Neural Networks (BRNNs) [32]. It is sometimes practical to take future inputs into account when making predictions with recurrent neural networks. This can be achieved by delaying the output when training a network; for example, using a delay of M with the training input/output pair $\mathbf{s}_{1:N,t}/z_{t-M}$. However, it was found [32] that if this delay is too large then it adversely affects the network prediction accuracy, thereby limiting how far into the future a network can account. An

alternative approach is to train an RNN with a reversed time series input, e.g., $\mathbf{s}_{N:1}$. This led to the Bidirectional Recurrent Neural Network in which two RNNs – one processing the input series forward in time and one backward in time – are connected to the same output layer. This architecture yields greater prediction accuracy as it predicts based on past inputs as well as future inputs.

Several existing classification methods have been discussed in this chapter. Of these, the Nearest Neighbor, Feedforward Neural Network (Multilayer Perceptron), and Bidirectional Long Short Term Memory Networks have been identified as the most promising for finding the posterior probability established in Chapter 3. These methods will be empirically evaluated and analyzed in Chapter 5.

Chapter 5

EXPERIMENTS

5.1 Experimental Setup

In order to evaluate how well these methods can approximate the posterior probabilities, we collected two sets of data with which to perform experiments. The first set was generated from real-world sensor data collected by an autonomous vehicle in Singapore. The second set was generated by the SUMO traffic simulator [20]. The methodology behind this data collection is discussed in this chapter, along with the architecture of all classifiers used in experiments.

5.2 Real Data Collection

As this work is targeting autonomous vehicle applications, it is a priority to test against real-world data collected by an autonomous vehicle. While synthetic data sets are suitable for a proof of concept, there is an unknown factor in regards to whether a system will work as designed in a real environment, especially so with a safety-critical system. Towards that end, we collected data with the Shared Computer Operated Transport autonomous vehicle platform at the Singapore-MIT Alliance for Research and Technology's (SMART) Autonomous Vehicles lab.

Data collection was performed in Singapore, at 56 intersections within the vicinity of the National University of Singapore campus. Spatial point cloud data was collected with a SICK LMS 151 LIDAR sensor operating at 50Hz. As there is no ground truth available with which to form the vehicle time series, the data was fed through a two-stage vehicle tracking algorithm.

Figure 5.1: The autonomous vehicle that performed data collection.

The first stage [33] decomposes the point cloud data into a subset of clusters, in which each cluster consists of a collection of points in close proximity to each other. The clusters are then tracked over several measurement frames to yield an average spatial position and a velocity vector for a given point in time.

The second stage treats these independent measurements as observations to a particle filter-based multi-target tracking algorithm [31]. In this algorithm, particle filters are used to model distinct vehicle tracks. Each new observation is associated with the particle filter that has the highest likelihood of producing that observation. Vehicle time series are then derived from the particle filters and down-sampled to 10Hz. Supervised labels were manually generated from camera inputs collected simultaneously with the LIDAR data. This process yielded a data set consisting of 1011 unique time series.

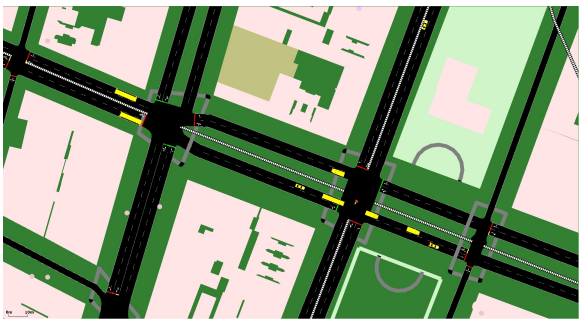

Figure 5.2: The SUMO road network for an intersection in New York City.

5.3 Simulated Data Collection

Despite the inherent value of real-world data, there is a limit to how much can be feasibly collected. Additionally, due to practical constraints, we could only collect data from nearby intersections which limits how well we can generalize. Therefore, we turned to synthetic data generated with the SUMO traffic simulator [20]. Road networks for 13 intersections were generated from OpenStreetMap data: 3 in Tempe, Arizona, 2 in New York City, New York, and 8 in Singapore. An example of an intersection in New York is shown in Fig. 5.2.

Simulations were run in which traffic passed through the intersections from each direction and either traveled straight, turned left, or turned right. The vehicles were uniformly distributed to one of three behavior models: aggressive, average, and submissive. These behaviors were manually defined and adjusted user-facing parameters. Table 5.1 lists the parameters for all behaviors.

SUMO is capable of writing floating car data (FCD) output, which contains the position, velocity, and heading of every vehicle at each sampling interval for the duration of the simulation. To correspond with the real data set, the sampling interval was fixed to 10Hz. This data was then transformed with respect to a chosen target vehicle, and used to generate state vectors for each other vehicle within a 50m sensor

Parameter	**Aggressive**	**Average**	**Submissive**
Acceleration	$3.0m/s^2$	$1.0m/s^2$	$1.0m/s^2$
Deceleration	$7.0m/s^2$	$5.0m/s^2$	$4.0m/s^2$
Speed Factor	1.5	1.0	0.9
Speed Deviation	0.3	0.2	0.2
Minimum Gap	$1.0m$	2	3
σ	1.0	0.5	0.5
τ	$0.5s$	$2.0s$	$3.0s$
Maximum Speed	$80m/s$	$70m/s$	$60m/s$
Impatience	1.0	0.5	0.1

Table 5.1: SUMO parameters for vehicle behaviors. The current speed is obtained by multiplying the road's speed limit by a sample from a normal distribution centered at Speed Factor with a standard deviation of Speed Deviation. Minimum gap is the following distance between a vehicle and its leader. Sigma is the variability in a vehicle's behavior. Tau is the following time between a vehicle and its leader, e.g. 3s. Impatience is the vehicle's disposition to forcibly changing lanes in front of others.

range of the target. Since the FCD data includes a vehicle identifier, these states can then be assembled into a time series for each vehicle. These time series' were segmented in order to coincide with the states of the intersection's traffic light and labeled as either *green* or *red*. This process yielded a data set consisting of 2311 unique time series.

5.4 Classifier Architecture

The BLSTM network used in these experiments is composed of an input layer followed by two parallel LSTM layers with 32 nodes each; one layer processes the input sequence forward and one layer backwards. The output from the LSTM layers is concatenated into a dropout layer with a 0.5 drop rate. The FFNN is a standard multilayer feedforward network with 3 hidden layers and 256 nodes per layer. In both networks, the size of the input layer is dependent on the vehicle state, while the output layer always consists of two nodes in order to produce a one-hot encoding of z_t. The networks are trained using RMSProp backpropagation with categorical cross-entropy loss and a softmax activation function. The K-Nearest Neighbor algorithm was evaluated for $K = 1$.

5.5 Results and Discussion

The first experiment of interest is to evaluate the relative performance of each classifier on our data sets. The classification accuracy is evaluated for the posterior probabilities produced by both the FFNN and BLSTM classifiers, with a train/validation/test set split of 60%/20%/20%, as well as the 1-NN classifier with a 80%/20% train/test split. This experiment reveals that despite being the simplest, the 1-NN classifier performs significantly better than all other classifiers on the *Real* data set with a 97% classification rate. This is an unexpected result, and interested in whether the noise reduction caused by the Bayesian tracking had a significant impact on the 1-NN performance, we created a *Real (No Track)* data set with only the raw measurements obtained by the clustering algorithm. However, despite slightly reduced performance, the 1-NN classifier is still the best performer on this data set. Results for BLSTM and feature sets containing acceleration are not included for this

Classifier	**Feature Set**	**Real** (No Track)	**Real**	**Sim**	**Sim-Real**
1-NN	x, y	0.678	0.853	0.737	0.719
1-NN	$x, y, \dot{x}, \dot{y}$	0.910	0.977	0.968	0.934
1-NN	$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$		0.943	0.972	0.969
FFNN	x, y	0.669	0.697	0.655	0.690
FFNN	$x, y, \dot{x}, \dot{y}$	0.850	0.897	0.796	0.774
FFNN	$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$		0.899	0.862	0.852
BLSTM	x, y		0.655	0.764	0.683
BLSTM	$x, y, \dot{x}, \dot{y}$		0.790	0.870	0.740
BLSTM	$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$		0.782	0.908	0.804

Table 5.2: The mean test accuracy for 1-Nearest Neighbor, Feedforward Neural Network, and Bidirectional Long Short Term Memory classifiers. The best classifier for each data set is highlighted in green.

data set as they require the time series information provided by the tracking algorithm.

Furthermore, 1-NN has the highest classification accuracy on the *Sim*ulation data set. This seems to indicate that the *Sim* data set is a good approximation of the real data set since it yields similar results, but on further analysis the test sample distribution between the two data sets is strikingly different. This can be observed in Fig. 5.3. The *Real* data set is heavily skewed, with a large portion of the test samples coming from intersections where only a small number of vehicles were observed for a short period of time. Meanwhile, the *Sim* data has a much flatter distribution over a wider domain.

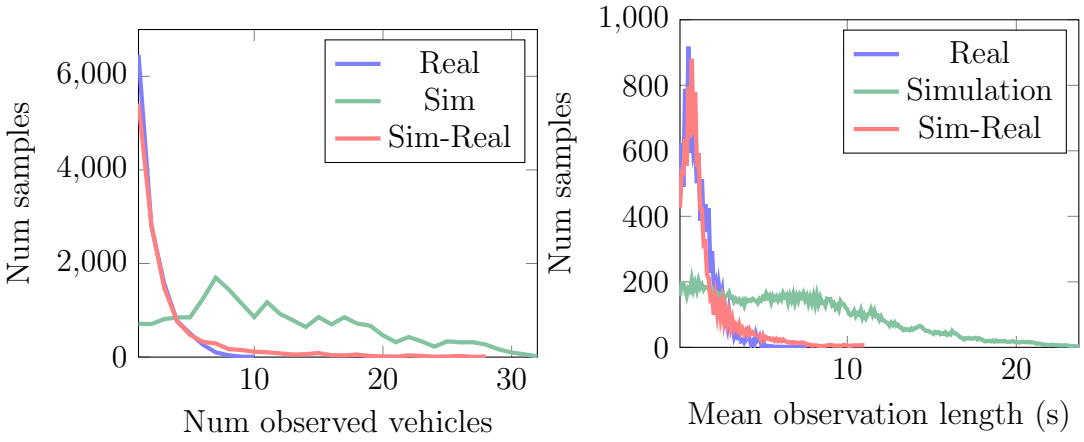

Figure 5.3: The distribution of test samples in each data set according to the number of observed vehicles per time step, and the mean observation length among all observed vehicles per time step.

In order to determine whether this distribution has a prominent effect on classification accuracy, we ran an optimization algorithm to minimize the Kullback-Leibler divergence between the distributions of the two data sets. This was accomplished by truncating a random portion of the time series by a variable factor. The initial KL divergence between the *Real* and *Sim* data sets is 1.35, however, after several rounds of this optimization routine that was reduced to 0.09. The resulting data set is referred to as *Sim-Real*, and it can be seen in Fig. 5.3 that the associated test sample distribution is similar to that of the *Real* data set. As in the other data sets, the 1-NN classifier is again the best performer on the *Sim-Real* data and indicates robustness to changes in the test sample distribution. Additionally, since the simulation data yields similar results to the real-world data and is capable of closely approximating the real-world observation distribution, we consider it an accurate representation of the real-world data.

The only time we observed the 1-NN classifier perform poorly is on data sets with a considerable amount of noise. Gaussian noise with a standard deviation of 2.0 was

Classifier	**Feature Set**	**Sim** (Noisy)	**Sim-Real** (Noisy)
1-NN	x, y	0.813	0.599
1-NN	$x, y, \dot{x}, \dot{y}$	838	0.648
1-NN	$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$	0.833	0.627
FFNN	x, y	0.642	0.684
FFNN	$x, y, \dot{x}, \dot{y}$	0.692	0.716
FFNN	$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$	0.695	0.709
BLSTM	x, y	0.765	0.679
BLSTM	$x, y, \dot{x}, \dot{y}$	0.874	0.742
BLSTM	$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$	0.863	0.718

Table 5.3: The mean test accuracy for 1-Nearest Neighbor, Feedforward Neural Network, and Bidirectional Long Short Term Memory classifiers. The best classifier for each noisy data set is highlighted in green, while the classifiers that are not significantly different are highlighted in yellow.

applied to all values in the *Sim* and *Sim-Real* data sets, resulting in *Noisy* variations. The results in Table 5.3 show that 1-NN yielded a considerably worse classification accuracy in this scenario, while BLSTM was largely unaffected by the additional noise and achieved the best accuracy with 87% and 74% on the *Sim* and *Sim-Real* data sets respectively.

The results in Tables 5.2 and 5.3 also allow us to examine the impact of the different vehicle states defined in Eqs. (3.1), (3.3), and (3.4) on the overall accuracy. The first observation we can make is that the addition of velocity information into the feature set results in a statistically significant (p-value < 0.05) increase in accuracy

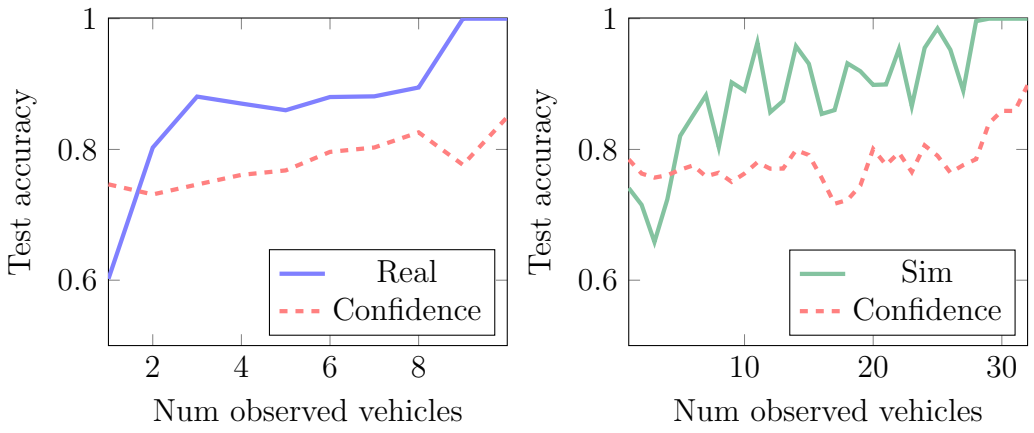

Figure 5.4: The BLSTM (full feature set) test accuracy for real data (solid blue line) and simulation data (solid green line) at each time step in relation to the number of observed vehicles.

for every classifier on every data set. This is a strong result, and in line with the hypothesis that the introduction of velocity information will help alleviate the intersection ambiguity problem. However, it is interesting to note that the addition of acceleration information does not always lead to a further increase in accuracy. The noisy data sets, in particular, actually exhibit either a statistically significant decrease in accuracy or no change at all. This suggests that we can reduce the complexity of our classifiers without penalizing accuracy on noisy data sets by leaving acceleration out of the feature set.

The second experiment is designed to test how the BLSTM classifier would perform in a realistic scenario. In real-world use, we do not have access to the full vehicle time series in the data sets; we only have the vehicle observations that have occurred until the current time step. The network is first trained with the full vehicle time series from all but one of the intersections. With the remaining time series, time is treated as a discrete value and incremented in steps. At every time step, the network is used to estimate the mean temporal probability in Eq. (4.13) from the vehicle

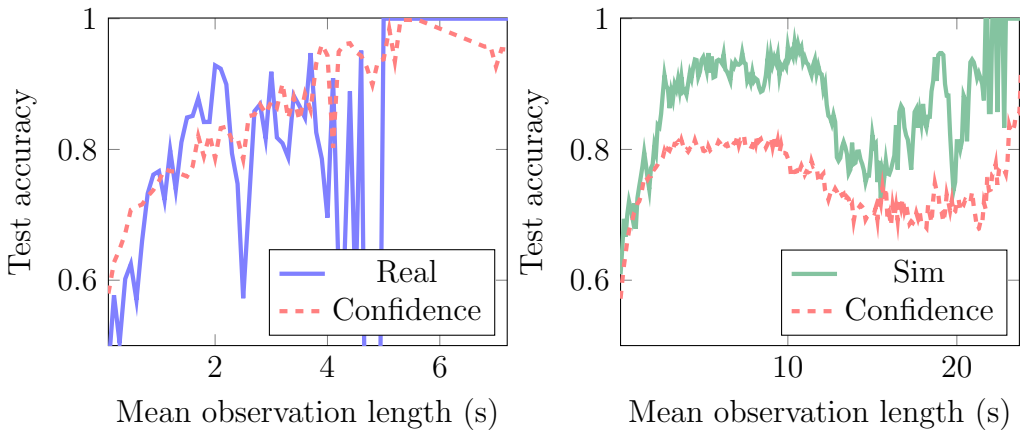

Figure 5.5: The BLSTM (full feature set) test accuracy for real data (solid blue line) and simulation data (solid green line) at each time step in relation to the mean observation length among all observed vehicles.

time series that have had an observation within the past 3 seconds. Only the observations that have occurred before the current time step are considered. The mean classification accuracy for all time steps is shown in Table 5.4.

With further analysis, it is evident that a significant number of misclassified time steps occur when only one vehicle is observed. As more distinct vehicles are observed, the classification accuracy increases, which is an intuitive result. This is visualized in Fig. 5.4. If we examine the distribution of test samples over the number of observed vehicles, it is clear that a large portion of the samples occur when only one vehicle is observable. Taking this into consideration, if the test accuracy is evaluated only for time steps in which two or more vehicles are observed, then the accuracy increases from 71% to 84% on the *Real* data set as shown in Table 5.4. In contrast, the simulation data has a flatter distribution, and as a result the corresponding accuracy does not see a proportional increase. With the positive correlation between accuracy and the number of vehicles, we might also expect such a relationship between the classification accuracy and the length of time that vehicles are observed. The plots

Feature Set	$\boldsymbol{\alpha}$	**Mean Real**	**Mean Sim**
$x, y, \dot{x}, \dot{y}$	1	0.693	0.861
$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$	1	0.718	0.866
$x, y, \dot{x}, \dot{y}$	2	0.837	0.863
$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$	2	0.842	0.870
$x, y, \dot{x}, \dot{y}$	3	0.877	0.868
$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$	3	0.876	0.876

Table 5.4: The BLSTM mean test accuracy for all time steps with at least α observed vehicles.

in Fig. 5.5 show a positive correlation, indicating that this is true to some extent.

Furthermore, there is also a positive relationship between the accuracy and the BLSTM classifier's prediction confidence, which we define as the maximum probability among all values of z_t. In other words, as the classifier observes more vehicles it grows more confident in the prediction and this results in a higher classification accuracy. However, there are instances in which this does not hold true. Specifically, it can be seen that the accuracy is poor while the prediction confidence is high for the *Real* data set when the mean observation length is between 4s and 5s in Fig. 5.5. On further analysis, this occurred when the target vehicle was stopped in front of pedestrians crossing a red light. A single vehicle was tracked for several seconds moving directly in front of the target vehicle with an average speed of $2.83m/s$. The most likely scenario is that the pedestrians crossing the street were mistaken for a vehicle turning left, which resulted in a high confidence prediction of a green light when in fact, the light was red.

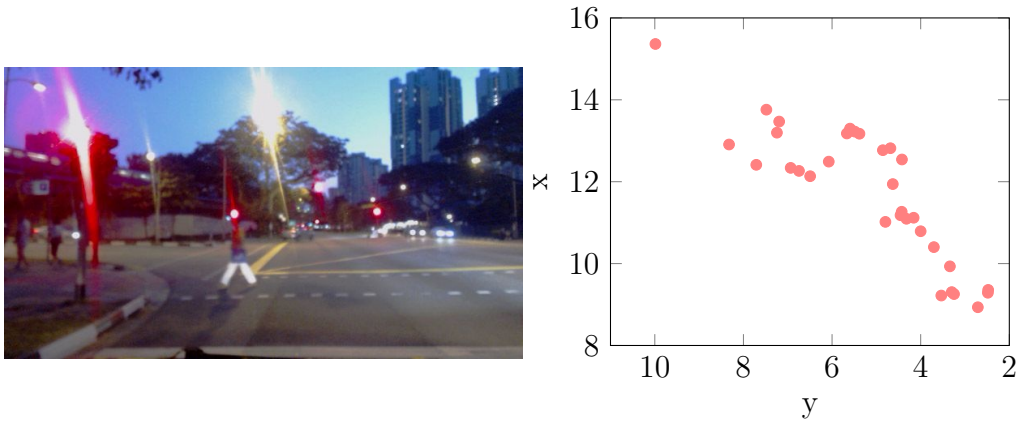

Figure 5.6: Scenario in which pedestrians mistaken for a vehicle result in a misclassification with high confidence. The camera image is on the left, and the corresponding time series given by the particle filter is on the right.

Chapter 6

CONCLUSION

This thesis has shown that it is possible to accurately infer the current state of a traffic light by analyzing the spatial movements of nearby vehicles with respect to a target vehicle. This method was evaluated on real-world data gathered in Singapore and synthetic data generated from a traffic simulator. In both cases, encouraging results were achieved with three different classifiers: a feedforward neural network, a bidirectional long short-term memory network, and a nearest neighbor classifier. It was found that in most tested scenarios, a nearest neighbor classifier obtained the best classification results (at the cost of higher computational and storage requirements). However, if the data is particularly noisy, better accuracy may be achieved with a BLSTM classifier.

Similar to a vision-based approach, the methodology presented here has failure cases in which inference produces wrong results. The most obvious case is when no vehicles are in observation range, however, it was also seen that in some scenarios more than one vehicle may need to be in observation range in order to make an accurate prediction. Likewise, there are specific instances in which the inference may be wrong if other vehicles are only observed for an extremely brief period of time. However, since the failure cases for our approach and a vision-based approach are not the same, we envision that the best use of our system is to combine it with a traditional vision-based method. Different failure cases suggests that the systems will complement each other, and result in a more robust detection system.

This work has also introduced the notion that synthetic data generated from a traffic simulator can be manipulated such that the KL divergence with respect to

another data set is minimized. The result is that the distribution of observations for the synthetic data closely aligns with that of another data set. We have used this approach to approximate a real-world data set, and conjecture that this can be extended to generate arbitrary distributions in order to test our method under varying traffic conditions. This also raises the interesting question of whether it is possible to train our classifier on a combination of real-world and synthetic data, and then employ this classifier in other real-world scenarios. We will examine these possibilities in future work.

REFERENCES

[1] Bengio, Y., P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", IEEE transactions on neural networks **5**, 2, 157–166 (1994).

[2] Bishop, C. M., *Neural networks for pattern recognition* (Oxford university press, 1995).

[3] Bishop, C. M., "Pattern recognition", Machine Learning **128** (2006).

[4] Campbell, J., H. B. Amor, M. H. Ang Jr. and G. Fainekos, "Traffic light status detection using movement patterns of vehicles", in "submitted to IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)", (IEEE, 2016).

[5] Campbell, J., C. E. Tuncali, P. Liu, T. P. Pavlic, U. Ozguner and G. Fainekos, "Modeling concurrency and reconfiguration in vehicular systems: A π-calculus approach", in "to appear in IEEE International Conference on Automation Science and Engineering (CASE)", (IEEE, 2016).

[6] Campbell, J., C. E. Tuncali, T. P. Pavlic and G. Fainekos, "Toward modeling concurrency and reconfiguration in vehicular systems", in "9th Interaction and Concurrency Experience, Satellite Workshop of DisCoTec 2016", (2016).

[7] Cover, T. and P. Hart, "Nearest neighbor pattern classification", IEEE transactions on information theory **13**, 1, 21–27 (1967).

[8] Elman, J. L., "Finding structure in time", Cognitive science **14**, 2, 179–211 (1990).

[9] Fairfield, N. and C. Urmson, "Traffic light mapping and detection", in "Robotics and Automation (ICRA), 2011 IEEE International Conference on", pp. 5421–5426 (IEEE, 2011).

[10] Funahashi, K.-I., "On the approximate realization of continuous mappings by neural networks", Neural networks **2**, 3, 183–192 (1989).

[11] Gradinescu, V., C. Gorgorin, R. Diaconescu, V. Cristea and L. Iftode, "Adaptive traffic lights using car-to-car communication", in "Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th", pp. 21–25 (IEEE, 2007).

[12] Graves, A., "Neural networks", in "Supervised Sequence Labelling with Recurrent Neural Networks", (Springer, 2012).

[13] Graves, A., "Sequence transduction with recurrent neural networks", arXiv preprint arXiv:1211.3711 (2012).

[14] Graves, A. and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures", Neural Networks **18**, 5, 602–610 (2005).

[15] Gupta, U., J. Campbell, U. Y. Ogras, R. Ayoub, M. Kishinevsky, F. Paterna and S. Gumussoy, "Adaptive performance prediction for integrated gpus", in "to appear in IEEE/ACM International Conference on Computer Aided Design (ICCAD)", (IEEE, 2016).

[16] Haykin, S. S., *Neural networks and learning machines*, vol. 3 (Pearson Education Upper Saddle River, 2009).

[17] Hochreiter, S. and J. Schmidhuber, "Long short-term memory", Neural computation **9**, 8, 1735–1780 (1997).

[18] Jordan, M. I., "Attractor dynamics and parallellism in a connectionist sequential machine", (1986).

[19] Kim, K., J. Campbell, W. Duong, Y. Zhang and G. Fainekos, "Discof+: Asynchronous discof with flexible decoupling for cooperative pathfinding in distributed systems", arXiv preprint arXiv:1506.03540 (2015).

[20] Krajzewicz, D., J. Erdmann, M. Behrisch and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility", International Journal On Advances in Systems and Measurements **5**, 3&4, 128–138 (2012).

[21] Levinson, J., J. Askeland, J. Dolson and S. Thrun, "Traffic light mapping, localization, and state detection for autonomous vehicles", in "Robotics and Automation (ICRA), 2011 IEEE International Conference on", pp. 5784–5791 (IEEE, 2011).

[22] Lindner, F., U. Kressel and S. Kaelberer, "Robust recognition of traffic signals", in "Intelligent Vehicles Symposium, 2004 IEEE", pp. 49–53 (IEEE, 2004).

[23] Maslekar, N., M. Boussedjra, J. Mouzna and H. Labiod, "Vanet based adaptive traffic signal control", in "Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd", pp. 1–5 (IEEE, 2011).

[24] Michel, O., "Webots: Professional mobile robot simulation", Journal of Advanced Robotics Systems **1**, 1, 39–42, URL `http://www.ars-journal.com/International-Journal-of-Advanced-Robotic-Systems/Volume-1/39-42.pdf` (2004).

[25] Park, J.-H. and C.-s. Jeong, "Real-time signal light detection", in "2008 Second International Conference on Future Generation Communication and Networking Symposia", pp. 139–142 (IEEE, 2008).

[26] Qin, B., Z. J. Chong, T. Bandyopadhyay, M. H. Ang, E. Frazzoli and D. Rus, "Learning pedestrian activities for semantic mapping", in "Robotics and Automation (ICRA), 2014 IEEE International Conference on", pp. 6062–6069 (IEEE, 2014).

[27] Richard, M. D. and R. P. Lippmann, "Neural network classifiers estimate bayesian a posteriori probabilities", Neural computation **3**, 4, 461–483 (1991).

[28] Rosenblatt, F., "The perceptron: a probabilistic model for information storage and organization in the brain.", Psychological review **65**, 6, 386 (1958).

[29] Rosenblatt, F., "Principles of neurodynamics. perceptrons and the theory of brain mechanisms", Tech. rep., DTIC Document (1961).

[30] Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation", Tech. rep., DTIC Document (1985).

[31] Schulz, D., W. Burgard, D. Fox and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association", in "Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on", vol. 2, pp. 1665–1670 (IEEE, 2001).

[32] Schuster, M. and K. K. Paliwal, "Bidirectional recurrent neural networks", Signal Processing, IEEE Transactions on **45**, 11, 2673–2681 (1997).

[33] Shen, X., S.-W. Kim and M. Ang, "Spatio-temporal motion features for laser-based moving objects detection and tracking", in "Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on", pp. 4253–4259 (IEEE, 2014).

[34] Tae-Hyun, H., J. In-Hak and C. Seong-Ik, "Detection of traffic lights for vision-based car navigation system", in "Advances in Image and Video Technology", pp. 682–691 (Springer, 2006).

[35] Werbos, P. J., "Backpropagation through time: what it does and how to do it", Proceedings of the IEEE **78**, 10, 1550–1560 (1990).

[36] Zhang, Y., K. Kim and G. Fainekos, "Discof: Cooperative pathfinding in distributed systems with limited sensing and communication range", in "Distributed Autonomous Robotic Systems", pp. 325–340 (Springer, 2016).