Moving Object Path Prediction for Traffic Scenes

Jaime Boanerjes Fernandez Roblero, B.C.Sc., M.C.Sc.

A Dissertation submitted in fulfilment of the

requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

Dublin City University

Faculty of Engineering and Computing, School of Computing

Supervisors

Dr. Suzanne Little

Prof. Noel E. O'Connor

January 2022

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Sign: [illegible] Student No.: 16212410 Date: 12/01/2022
(Jaime Boanerjes Fernandez Roblero)

Acknowledgements

I would first like to express my profound gratitude to my supervisor, Dr. Suzanne Little, whose expertise was invaluable in formulating this research work. Her insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. Her wise advice, for both professional and personal life, made easier this research journey. I will be ever grateful to my supervisor Prof. Noel E O'Connor for his brilliant ideas and for giving me an excellent opportunity to work under his supervision, for supporting my research directions and encouraging me throughout my PhD. I would like to thank my supervisor, Alan F. Smeaton for his excellent inputs, intriguing questions, and consistent encouragement. Thanks to the three of them for being a true inspiration.

I extend my sincere thanks to the Insight SFI Research Centre for Data Analytics and Science Foundation Ireland for funding my research. To the EU H2020 VI-DAS Project for funding my work and for the valuable experience gained during its development. I am indebted to the VI-DAS project team, for the amazing experience and all the knowledge I got while working with them.

I would like to recognize the invaluable assistance that all the Insight team provided during my study. For making my life easier.

On a personal note, I wish to acknowledge the support and great love of my family; my mother, Maria and my father, Jaime. They kept me going on and this work would not have been possible without them.

Finally, I could not have completed this dissertation without the support of all my friends, Carlos M., Mariana A., and all of you guys, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

I thank all my friends and colleagues in DCU, for their companionship. I would also like to extend my sincere thanks to everyone who helped and supported me throughout my PhD.

Table of Contents

List of Figures

List of Tables

List of Publications

1. *Peer-reviewed*: Fernandez, J. B., Little, S., O'Connor, N. E. (2019, November). A Single-Shot Approach Using an LSTM for Moving Object Path Prediction. In *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)* (pp. 1-6). IEEE.

2. *Peer-reviewed*: Fernandez, J. B.; Little, S. and O'Connor, N. E. (2020). Multiple Path Prediction for Traffic Scenes using LSTMs and Mixture Density Models. In *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems - Volume 1: VEHITS*, ISBN 978-989-758-419-0, pages 481-488. DOI: 10.5220/0009412204810488

Demos

1. *Peer-reviewed*: Fernandez, J. B., Venkatesh, G. M., Zhang, D., Little, S., O'Connor, N. E. (2019, September). Semi-Automatic Multi-Object Video Annotation Based on Tracking, Prediction and Semantic Segmentation. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)* (pp. 1-4). IEEE.

List of Abbreviations and Acronyms

ADAS	Advanced driver assistance systems
ADE	Average displacement error
ANN	Artificial neural network
API	Application programming interface
ATP	Absolute tracklet position.
BEV	Bird's-eye view
CNN	Convolutional neural network
CV	Constant velocity
FDE	Final displacement error
FPS	Frames per second
GAP	Global pooling layer
GMM	Gaussian mixture models
GPS	Global positioning system
GRU	Gated recurrent unit
HOG	Histogram of oriented gradients
IMU	Inertial measurement unit
KF	Kalman filter
LDM	Local dynamic map
LIDAR	Light detection and ranging
LSTM	Long short-term memory
MDM	Mixture density models
MDN	Mixture density networks
MLP	Multilayer perceptron
MSE	Mean squared error
NHTSA	National highway traffic safety administration
P.H.	Prediction horizon
ReLU	Rectified linear unit
RMSE	Root of the mean squared error
RNN	Recurrent neural network
RTP	Relative tracklet position
WSADE	Weighted sum of average displacement error
WSFDE	Weighted sum of final displacement error

Abstract

Jaime Boanerjes Fernandez Roblero

Moving Object Path Prediction for Traffic Scenes

Accurate and efficient inference and prediction are important elements in intelligent systems. Knowing in advance the behaviour of an entity, such as the price of a product in the future, the weather in the next few days or the position of an object in the near future, is important for several applications like stock market, weather forecasting, robotics and more recently for autonomous vehicles. The aim of this work is to investigate and develop a novel approach for predicting the path of moving objects such as pedestrians and vehicles in the context of ego-cameras, like those mounted on a vehicle or a person. Due to the sequential nature of the data presented in paths, Recurrent Neural Networks (RNNs) are exploited, specifically Long Short-Term Memory Networks (LSTMs), due to their ability to process this type of data. LSTMs have the limitation of only predicting a single path per tracklet. Path prediction requires predicting with a level of uncertainty. Predicting multiple future paths instead of a single one is therefore a more realistic manner of approaching this task. In this work, predicting a set of future paths with associated uncertainty was achieved by combining LSTMs and MDNs. One of the objectives of this work is to include more information than simple position in the path prediction task, such as velocity of the ego vehicle and contextual information of the surroundings. Though the main interest of this work is on egocentric cameras experiments were also conducted using fixed cameras for a surveillance perspective. Two public datasets were used: KITTI and CityFlow. In summary, this thesis extends moving object path prediction methods in the context of traffic scenes for objects such as pedestrians, vehicles, cyclists.

Chapter 1

Introduction

1.1 Chapter Overview

This chapter provides a general introduction to this thesis, presents the motivation and states the purpose of this research. Section 1.2 presents the motivation behind this research. Section 1.3 gives an overview of the problem of path prediction. Section 1.4. presents the main objective of the work. Section 1.5 Lists the hypotheses and research questions derived from the previously reported works on path prediction. Section 1.6 describes the research contributions. Finally, section 1.7 outlines the organisation and structure of this thesis.

1.2 Motivation

Road traffic collisions are an important cause of death and disability worldwide. Every year around the world 1.2 million people are killed and up to 50 million are injured or disabled as a result of road traffic collisions [9, 21, 82]. According to a recent technical report by the National Highway Traffic Safety Administration (NHTSA), 94% of road accidents are caused by human errors [123].

Autonomous vehicles are becoming a reality. Automobiles equipped with ADAS (Advanced Driver Assistance Systems) and sensors such as cameras, radars and LIDARs are now common place, as shown in the Figure 1.1. Many of the accidents on the road can be avoided or at least can be mitigated by acting seconds in advance [30]. For this reason, safety on the road is one of the main objectives in the development of ADAS. Predicting where a pedestrian or a vehicle will be in the near future in a scene, termed path prediction in this thesis, can provide useful information that allows for an ADAS to react in those seconds. Advances in computer vision and machine learning techniques can help to improve prediction accuracy, efficiency and speed. In this research, these new techniques will be leveraged to create a novel approach for path prediction in moving cameras.

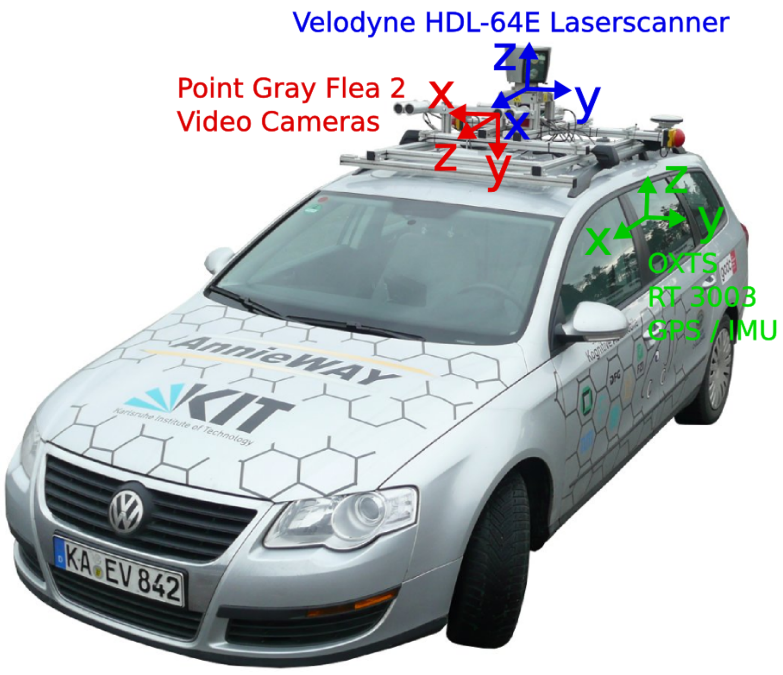


Figure 1.1: Sensors on a vehicle [26]

1.3 Path Prediction Overview

Advanced Driver Assistance Systems (ADAS) refer to those systems that provide aid to a driver at a certain level [35, 96]. For instance, at a low level of automation (e.g., Level 1 and 2), ADAS can warn a distracted driver that a pedestrian or a vehicle is approaching the front on the vehicle. At a higher level (e.g., Level 3 or 4), ADAS activates the brakes of the vehicle with the purpose of avoiding or mitigating the collision. In order to act, ADAS require awareness of the surrounding scene. For that reason, nowadays vehicles are also equipped with sensors such as cameras, radars and LIDARs. Amongst these, because of the low cost and the richness of information it can offer, the camera is becoming the most important sensor and a significant amount of research papers are based on visual information. Prediction is a task that refers to the process of knowing in advance the behaviour of an entity such as the price of a product in the future, the weather in the next few days or the position of an object in the near future. Predicting future behaviour is important for several application like the stock market [10], weather forecasting [20], robotics [24] and more recently for autonomous vehicles [41]. Autonomous navigation requires accurate and detailed models of the static and dynamic environment where the vehicle is moving. Significant advances has been achieved in scenarios free of moving (dynamic) objects such as in robotics. In contrast, environments with moving objects, for instance pedestrians cars, and cyclists, still pose significant challenges for navigation [37, 17, 7]. Techniques for vehicle and pedestrian detection have also made significant progress in recent years, see 1.2, and methods that allow for the detection of these are increasingly reliable [34, 23, 26]. Detection of moving objects is useful in order to be aware of the surroundings of a vehicle since this information cannot be captured by a static road map.

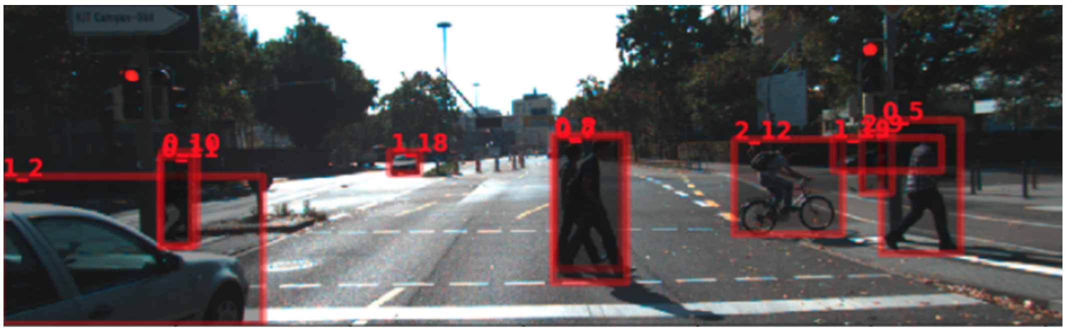


Figure 1.2: Object Detection and Tracking

Path prediction refers to predicting the possible trajectory that an object could follow in the future and thus determine its future location. Much research has been conducted on static cameras like those used in surveillance videos. However, working with cameras located on a mobile platform, for example on a vehicle, a robot or a person poses several challenges including the movement of the platform where the camera is mounted, dynamic background, shorter observed paths.

The approaches developed to address the problem of path prediction from a moving platform can be classified according to the type of information considered and the assumptions made when developing the approach. From the simplest to the most complex, the following classification approaches can be used: physical-based, manoeuvre-based and interaction-aware [41]. Physical-based approaches only take into account basic information such as the observed previous location of the object and its velocity. Manoeuvre-based approaches first classify the action that the object is likely to carry out, e.g. stopping, accelerating. Given this, these approaches assume that the next action will match the current manoeuvre. Finally, interaction-aware approaches take into account both the object as an isolated identity and also how its action and the action of other objects will affect its possible future path.

Several techniques are commonly used to process the data required in each approach, from the well-known Kalman filter [91] to more complex machine learning techniques. Most recently, deep neural networks are being used to capture and process all the information in an end-to-end architecture. The more information included in an approach the more reliable the result is likely to be.

My research is based on the premise that whilst knowing where an object is currently located is already useful, predicting its location in the future is of great importance for autonomous vehicles for safe and efficient driving. It is also necessary for many Advanced Driver Assistance Systems (ADAS) where both the trajectory of the ego vehicle (vehicle equipped with the ADAS and sensors) and the trajectory of other objects on the road have to be predicted. Whilst considered within the context of ADAS in this thesis, path prediction has many other applications, for example, in assistive technology such as navigation aids for blind people to avoid collision. Considerable research has been done in tracking moving objects or predicting a path using static or fixed cameras, e.g., in surveillance [58, 62, 69]. However, in this research the main focus is on egocentric, moving cameras, such as those mounted on a vehicle or a person. In addition, an holistic approach is targeted to incorporate more information in the prediction task – information such as the object type, ego-motion, a segmented map of the road to know where an object is located or data about its surroundings in the prediction task. Finally, different to several works on intent or behavior prediction, which can be modeled as classification problems, i.e. crossing, stopping, our main objective is to predict future path (x, y) positions for the target object, which is a regression problem. In this research the term tracklet ,tr, as explained in 2.2.4 Tracklets, will be used to refer to the past or prior spatial trajectory of a moving object as in [99] and [85].

1.4 Objective

The objective of this research is to develop a novel technique to accurately predict the Average Displacement Error (ADE) and Final Displacement Error (FDE) of the probable path of moving objects, such as pedestrians and vehicles, based on data from egocentric cameras from a moving vehicle and incorporating contextual map data.

1.5 Hypotheses and Research Questions

The hypothesis of this research is specified as follows: *LSTMs are an effective tool for path prediction and existing work can be extended to predict multiple paths and to include contextual information, creating a holistic approach leading to improved performance in terms of ADE and FDE.*

To investigate the hypothesis, the following questions are considered:

- **Q1 How should the observed object position (tracklets) be best represented?** In this research, the prediction of the future path of a moving object is based on its past observed path (tracklets) along with more information of the scene (context) where this prediction is happening. This first research question aims to investigate how positional information can be best represented to be fed to our LSTMs-Based approach. RQ1 question was analysed in chapter 2, and explored and evaluated in chapter 3. From chapter 2 and 3 it was evidenced that representing the position of objects as time series and as Relative Tracklet Position (RTP) was suitable to predict the future path of an object. Furthermore, in chapter 6 was concluded from section 6.7 Exploration of multimodal features that representing the object position first in a latent space led to error reduction.

- **Q2 How can LSTMs be extended to predict multiple paths?** Predicting a set of paths with associated uncertainty is a more realistic way of predicting the future position of objects instead of a single one. LSTMs are only able of predicting a sigle path per observed tracklet, because of that this research question aims to explore a way to extend LSTMs to predict a set of paths. RQ2 is initially explored in chapter 4, where a study of several variants of LSTMS was performed on predicting the future path of objects in traffic scenes, this with the objective of understanding their behaviour and the way in which each variant processes the input sequential data. RQ2 was also studied in chapter 5 where LSTM architectures are successfully used along with MDN (Mixture Density Networks) for predicting a set of paths per observed tracklet along with its associated uncertainty.

- **Q3 Are Long Short-Term Memory (LSTM) architectures suitable for sequential and enriched trajectory information?** LSTMs have shown good performance when dealing with sequential information such as time series. However, this research aims to include contextual information such as visual features, ego-vehicle information, other objects position leading this way to a holistic approach. The purpose of this RQ3 is to evidence that LSTMs are able to process all these type of information. The initial exploration of LSTM architectures was done in chapter 3, where a single-shot approach was developed using object position only. RQ3 is explored in chapter 5 and more deeply in chapter 6. In chapter 5, an initial exploration of adding more features to the tracklets was done showing that these extra features improved the overall performance of the model. Finally, in chapter 6, a more complete exploration was done by adding additional contextual features to the tracklets. Starting from only using object position to using visual-object (images) features, ego-vehicle information, other object position, and scene-image features different combinations were evaluated. The results showed that LSTMs models are able to process sequences of enriched trajectory information.

- **Q4 How can contextual information of a scene be used to improve path prediction results over only using x, y positional features?** How to process/fuse the available features inside a model is an important point when trying to use different type of information. The fact that a set of features does not lead error reduction does not mean this set is not working, the reason could be the way this set of features are being fused inside a model. This research question aims to find a way to best fuse contextual features for the path prediction task. RQ4 was deeply explored in chapter 6 where additional features describing the environment were included using different fusion strategies. Three fusion strategies were evaluated – Early fusion of raw features, Early fusion of latent space features, and Middle fusion of latent space features. An interesting observation here is that both the information available and the fusion method are highly important. This was observed in each combination where using middle fusion on latent space features leads to better performance.

1.6 Research Contributions

The novel contributions of this work are:

1. Thorough evaluation of different variants of LSTM models to understand their behaviour and performance on the task of predicting the future path of three different objects – pedestrians, vehicles and cyclists on four prediction horizons.

2. Proposing the use of LSTMs with MDN to predict multiple paths with associated uncertainty.

3. Extensive exploration combining several contextual features in traffic scenarios to assess their impact in the path prediction task against only using x, y positional information.

4. Exploration of different fusion strategies to evaluate their performance gain in the overall prediction task.

5. Proposing an end-to-end architecture to represent and fuse contextual information normally present in traffic scenarios to predict the path of moving objects using LSTM and CNN architectures.

6. Improvement of the performance of individual models in path predictions by building ensembles.

1.7 Thesis Outline

The overall structure of the thesis takes the form of seven chapters organised as follows.

Chapter 2 describes the necessary technical background and presents a comprehensive summary of the related work on the field of path prediction. The chapter begins by laying out the theoretical dimensions of the research with some important preliminary concepts such as a formal definition of tracklets, semantic information, local dynamic map. The chapter also analyses the state-of-the-art works presented in the literature, and finally the chapter presents a description of the datasets and the evaluation metrics used in this research.

Chapter 3 presents the evaluation of the performance of two baseline methodologies specifically the Kalman Filter and the vanilla LSTM. This chapter also presents the experimental methodology followed to process the data in order to be able of predicting a future path.

Chapter 4 presents an evaluation study of several RNNs variants on the path prediction task. The chapter begins by explaining the motivation of this study. It then goes on to a description of the RNNs variants used. Finally, the evaluation performance of the models is presented along with a discussion on the results obtained.

Chapter 5 describes the use of Mixture Density Networks for prediction of multiple paths. The chapter begins by explaining how to overcome the problem of only predicting one path per tracklet. The following part explains the model architecture used along with the process followed to extract the multiple paths from the output of the network. It finishes by presenting the evaluation results.

Chapter 6 presents a novel approach to path prediction using contextual features to enrich the training information for path prediction. It starts by stating the reason for using enriched tracklets. The contextual features used are then explained. Following this, the methodology is discussed along with the objective model to be developed. Finally, the results obtained along with a detailed discussion is presented.

Chapter 7 concludes this thesis by summarising the research methodology, discussing the solutions to the research questions, highlighting the contributions, and providing future directions of research based on the learnings arising from the research carried out.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter describes the technical background of this research and presents a comprehensive summary of the related work in the field of path prediction. The chapter begins by laying out the theoretical dimensions of the research with some important preliminary concepts such as a formal definition of *tracklets*, *semantic information*, *local dynamic map* and other key terms in Section 2.2. Section 2.3 describes important techniques in path prediction. Section 2.4 analyses the state-of-the-art work presented in the literature. Section 2.5 describes the available datasets. Section 2.6 describes the evaluation metrics used in this research and Section 2.7 discusses the overall state of path prediction from moving cameras. Finally, in Section 2.8 a summary is given.

2.2 Preliminary Concepts

2.2.1 Ego vehicles

Ego vehicle or ego car is a term that is widely used in intelligent transportation and autonomous vehicles literature to refer to the vehicle that is carrying the sensors and any intelligent processing systems [123, 43]. The ego vehicle is the centre of attention and the scene is observed from its perspective. This research focuses on predicting the paths of the moving objects around an ego vehicle and not on predicting the path of the ego vehicle itself.

2.2.2 Egocentric Cameras

The reduction in size of cameras allows for them to be carried or installed in almost any platform such as a wearable on a person or on a mobile platform like a vehicle or drone [18], Figure 2.1. This type of camera provides a first-person point of view recording and is the primary type of data used in this research. This is in contrast to fixed cameras used in applications such as surveillance that operate from a single view point perspective. Working with egocentric cameras poses a challenge to the task of path prediction due to the ego-motion. [73] define ego-motion as "the camera's motion within an environment, relative to a rigid scene, where the motion can be 3D".

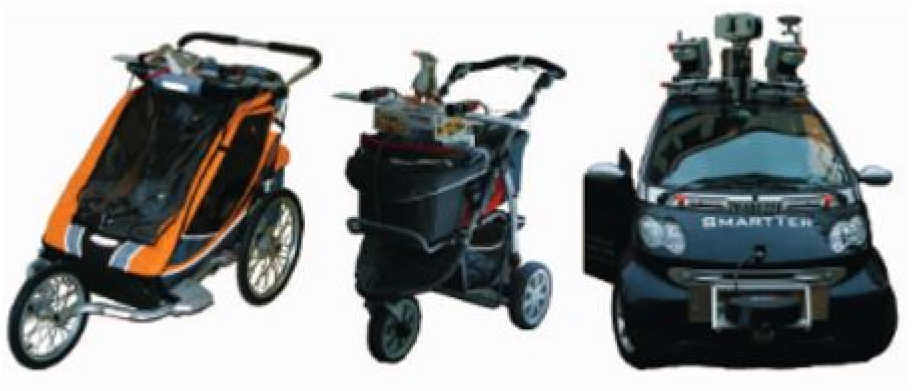


Figure 2.1: Mobile platforms with ego-centric cameras [18]

2.2.3 Holistic Approach

When addressing a problem, many approaches focus only on one source of information when trying to find a solution. In the path prediction research literature, even when they use different approaches such as the Kalman Filter [33, 40], or LSTM architectures such as Vanilla LSTMs [66], Stacked LSTMs [79, 114], and Encoder-Decoder Architectures [94] they base the prediction task on only positional information. However, a holistic approach will try to solve this problem by taking into account all available information on a scene. In this research, a holistic approach is used to develop a novel path prediction technique by including information such as the visual information of objects, ego-vehicle features, position of other object on the scene and its surroundings rather than only considering the position of the moving object in isolation.

2.2.4 Tracklets

A spatial trajectory is a trace generated by a moving object in space [56], usually represented by a series of chronologically ordered points, p, $p1$, $p2$, $\cdots$, pn, where each point consists of a spatial coordinate set and a time stamp:

$$p = (x, y, t) \tag{2.1}$$

where x, y are Cartesian coordinates and t indicates the point in time or timestamp (absolute, relative or frame number). In this research the term tracklet ,tr, is used to refer to the past or prior spatial trajectory of a moving object as in [99] and [85]. Example of complete trajectories (composed of tracklets) in different contexts are depicted in Figure 2.2 and Figure 2.3.

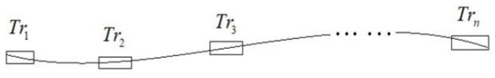


Figure 2.2: Trajectory generated by GPS tracking devices [86].

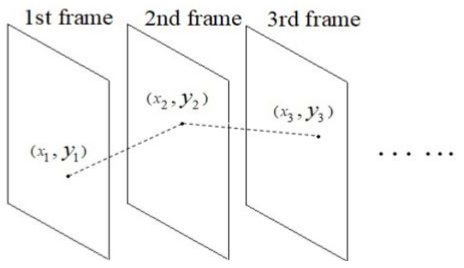


Figure 2.3: Trajectory generated from camera device [86].

2.2.5 Path as Time Series

A path P is a set of tracklets, tr, that contains information such as $tr(x, y)$ position (coordinates) of an object that travels a given space, $P = \{tr_{t1}, tr_{t2}, ..., tr_{tlength}\}$. Each tr is a measure given for a sensor in intervals of time and in an ordered manner, $tr(x, y, time)$. This means that a path is a sequence of measurements of the same variable collected over time, where the order matters, resulting in a time series. Because of this, a path can be seen as a multivariate time series that has two time-dependent variables. Each variable depends on its past values, and this dependency is used for forecasting future values. So the task of path prediction can be seen as forecasting a multivariate multi-step time series, Figure 2.4.

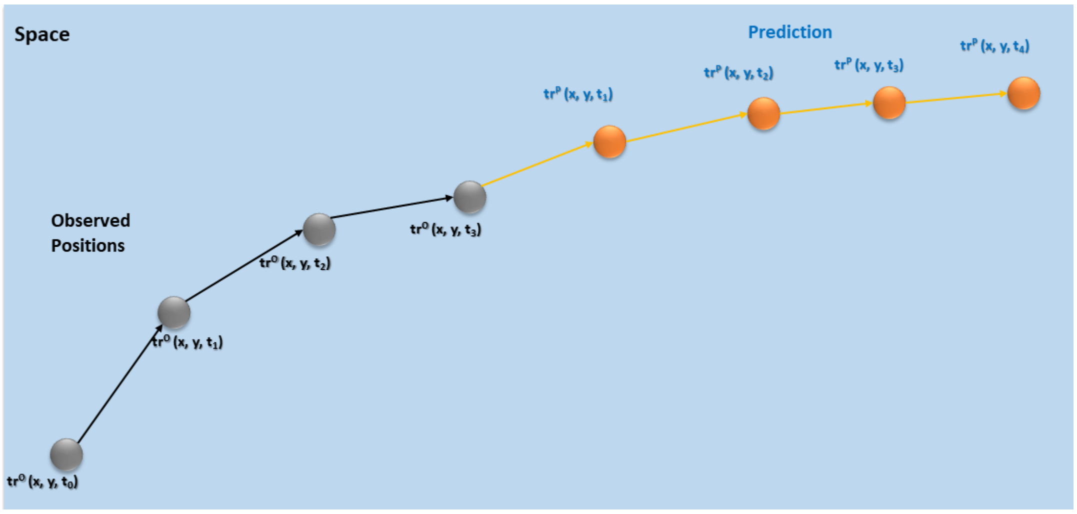


Figure 2.4: Path as time series prediction

2.2.6 Sliding Window

Looking at a path as time series data, it can be processed with the sliding window method to extract the observed tr^O and ground truth tracklet tr^G segments that can be used to train a sequential model. This method consists in splitting the sequence data in fixed-length segments by a fixed-length time window that advances one step at time along the sequence, as depicted in Figure 2.5. tr^O is the input data to fit the model and tr^G is the target output of the model.

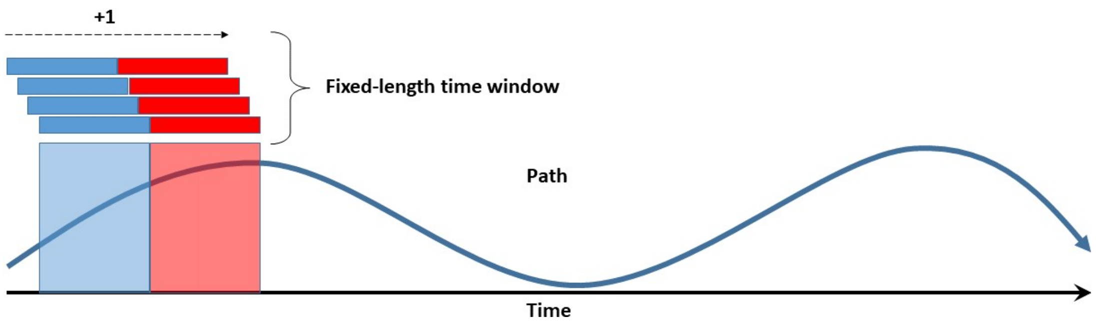


Figure 2.5: Using sliding window to construct supervised learning examples from a path seen as time series data. Blue: observed tracklet. Red: ground truth tracklets.

2.2.7 Image Coordinates and 3D Information

An image coordinate is the position of the object in the 2D plane of an image. These are given in pixels as (x, y) points. KITTI, one of the data sets used in this research (see Section Traffic Datasets), gives the bounding boxes of the objects in this plane as (x_1, y_1, x_2, y_2).

3D information refers to the position of the objects in the real world with respect to the camera. KITTI provides this information as the position in x, y, z along with the dimensions of each object height, width and length – (x, y, z, h, w, l). These measurements are given in metres. In this research, this information was leveraged to create a "birds-eye view" (BEV) of the objects to consider their positions in 3D space. The BEV is created as it gives a better understanding of the position of the objects on the Z axis (depth positional information), and the X axis(lateral position). In other words, it gives the positional information of the objects standing on the ground. Fig. 2.6 depicts the objects of a selected frame in both image coordinate and BEV with the object class label number and object identifier number.

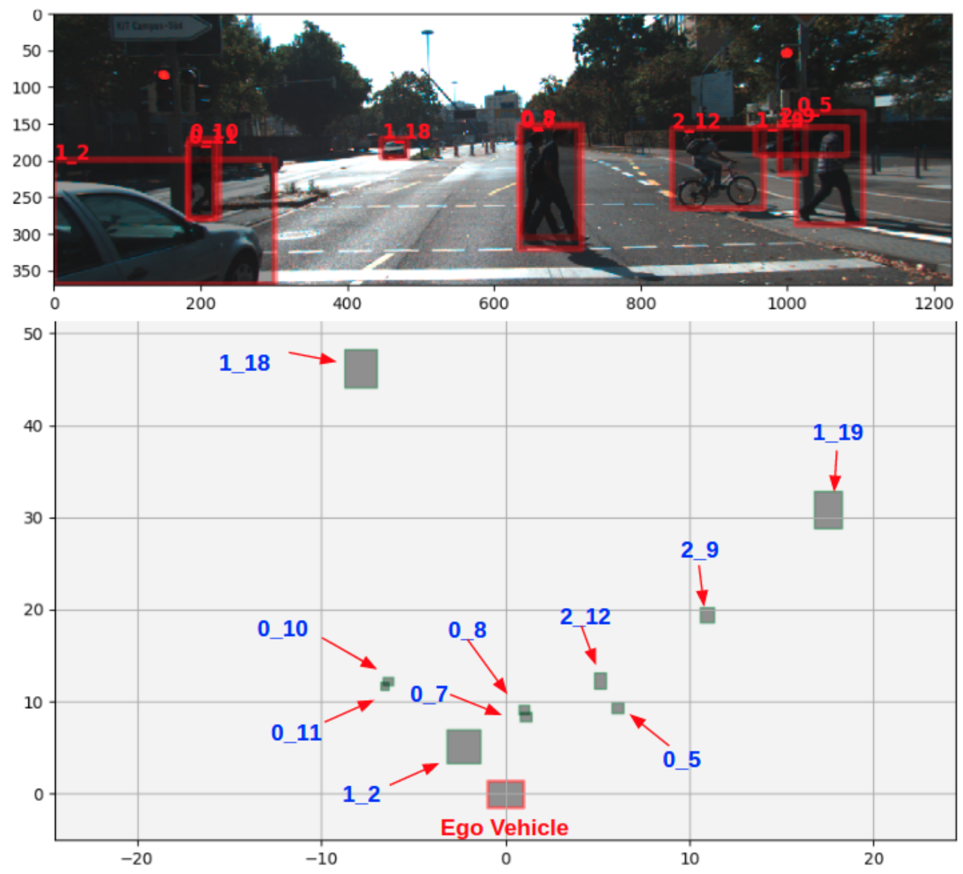


Figure 2.6: Image coordinate (top) and bird's-eye view (bottom) perspective with the class label and object identifier.

2.2.8 Semantic or Contextual Information

Semantic or contextual information refers to all types of information about the surrounds of the ego vehicle and the object whose path is being analysed and predicted. The increasing availability of sensors on vehicles and the technology to process the data coming from them, enables a range of information to be reliably detected and classified. This information includes [103]:

- **Static:** road lanes, traffic lights, signs and buildings..
- **Temporary:** weather, object velocity, orientation.
- **Dynamic:** pedestrians, cyclists, other vehicles in a scene.
- **Geographical and Local context:** GPS, road map, urban/rural.

In a scene, when trying to predict the future path of an object, all these types of information that surround an ego vehicle can be used together as a whole to create a more robust approach (holistic) that leverage the relationship of all these features.

2.2.9 LDM (Local Dynamic Map)

LDM, which is now being standardized in Europe, is an aggregation of data for use by cooperative ITS (Intelligent Transport Systems) [53]. It adopts a four-layer model as shown in Figure 2.7. The first or bottom layer consists of static data such as road data, the second layer consists of static data such as signals not included in map data, the third layer consists of data such as congestion and other traffic conditions, and the fourth or top layer consists of dynamic data such as automotive sensor information [53]. In this work we focus on using information that belong to the fourth layer, which is information obtained by means of sensors such as cameras.

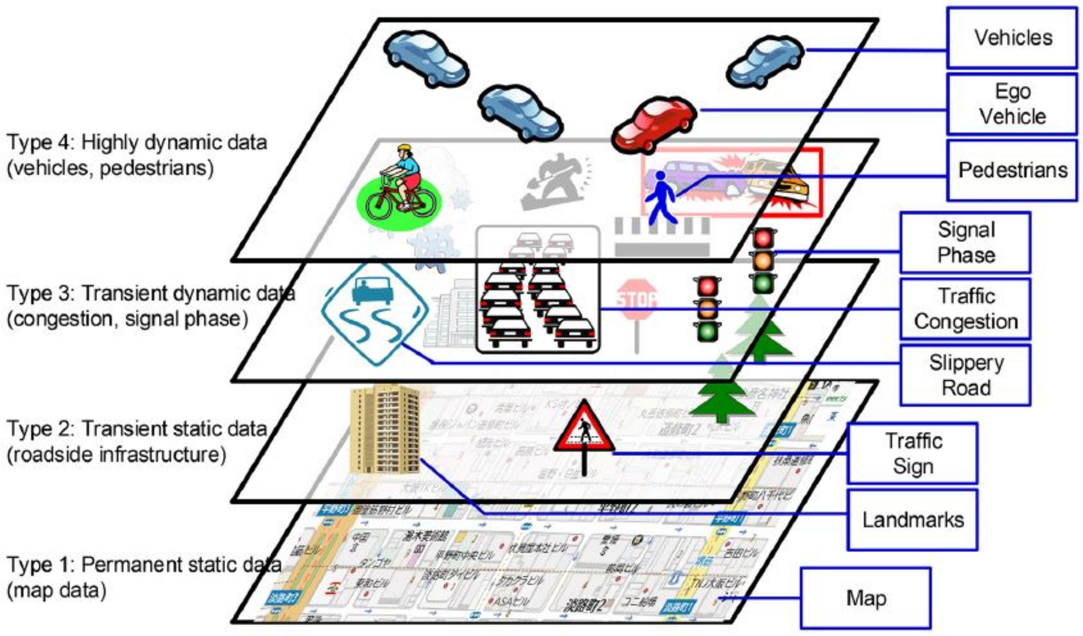

Type 4: Highly dynamic data (vehicles, pedestrians)
Type 3: Transient dynamic data (congestion, signal phase)
Type 2: Transient static data (roadside infrastructure)
Type 1: Permanent static data (map data)
Vehicles
Ego Vehicle
Pedestrians
Signal Phase
Traffic Congestion
Slippery Road
Traffic Sign
Landmarks
Map

Figure 2.7: The four layers of the LDM [53].

2.2.10 Sequential and Enriched Tracklet Information

Sequential information, such as time series or a path created by the spatial location visited by an object, have an important characteristic: they are time dependant. In other words, they are a sequence taken at successive equally spaced points in time and have an order. Considering a path specifically, this is a set of tracklets ,$P = \{tr_{t1}, tr_{t2}, ..., tr_{tlength}\}$, representing the past location of an object in a given space. These paths normally are characterised by $tr(x, y, time)$ position at a given time. However, with the availability of semantic and contextual information these tracks can be enriched by adding more features such as the orientation, velocity or where the object is located in the scene, and its surroundings. Enriched tracks represent the fusion of different sources of information. A challenge in getting this fusion of information is the synchronization of all different sources. At a sensor level, the synchronization is difficult due to the the timestamp (frequency) at which each sensor sample a signal from a scene. At a feature level, as these features are of different types and are not normally used together, they have to be first linked to belong to the same timestamp in a scene.

2.3 Important Techniques in Path Prediction

This research work makes use of some existing approaches in the literature of path prediction and computer vision to create an holistic path prediction approach able to input and fuse different type of information. The exploration started from the Kalman filter that help us to understand the processing of sequential x, y positional information (trackelts) as time series and predict the next position in the sequence. Then, this led us to explore more complex architectures called RNNs (Recurrent Neural Networks) specifically Long Short-Term Memory architectures (LSTMs) which is a neural network able to process sequential data taking into account long and short dependencies in the sequence to predict the next n future positions. While exploring LSTMs it was found that they have the limitation of only able to predict one single path per observed tracklet, so in the search for facing this limitation MDNs (Mixture Density Networks) were found to be suitable to be used along with LSTMs. This way, the LSTMs gives memory to the input network to process sequential data and the mixture model contributes to produce multiple paths with associated uncertainty. Finally as the objective is to create a holistic approach able to process different type of information including visual information, Convolutional Neural Networks (CNNs) are used as feature extraction techniques. These techniques are described in detail in the next sections.

2.3.1 The Kalman Filter

The Kalman filter is a traditional approach widely used for predicting the position of an object in the near future. The Kalman filter, based on the measurements given about the position of a given object, is able to modify its internal parameters during operation and according to that predict the future position, see Figure 2.8. The Kalman filter model predicts the status of a system according to the previous state at time $t-1$ [25]:

$$x_t = F_t x_{t-1} + B_t u_t + w_t \tag{2.2}$$

where

- x_t is the state vector containing the terms of interest for the system (e.g., position, velocity, heading) at time t.
- u_t is the vector containing any control inputs (steering angle, throttle setting, braking force).
- F_t is the state transition matrix which applies the effect of each system state parameter at time $t-1$ on the system state at time t (e.g., the position and velocity at time $t-1$ both affect the position at time t).
- B_t is the control input matrix which applies the effect of each control input parameter in the vector u_t on the state vector (e.g., applies the effect of the

throttle setting on the system velocity and position).

- w_t is the vector containing the process noise terms for each parameter in the state vector. The process noise is assumed to be drawn from a zero mean multivariate normal distribution with covariance given by the covariance matrix Q_t.

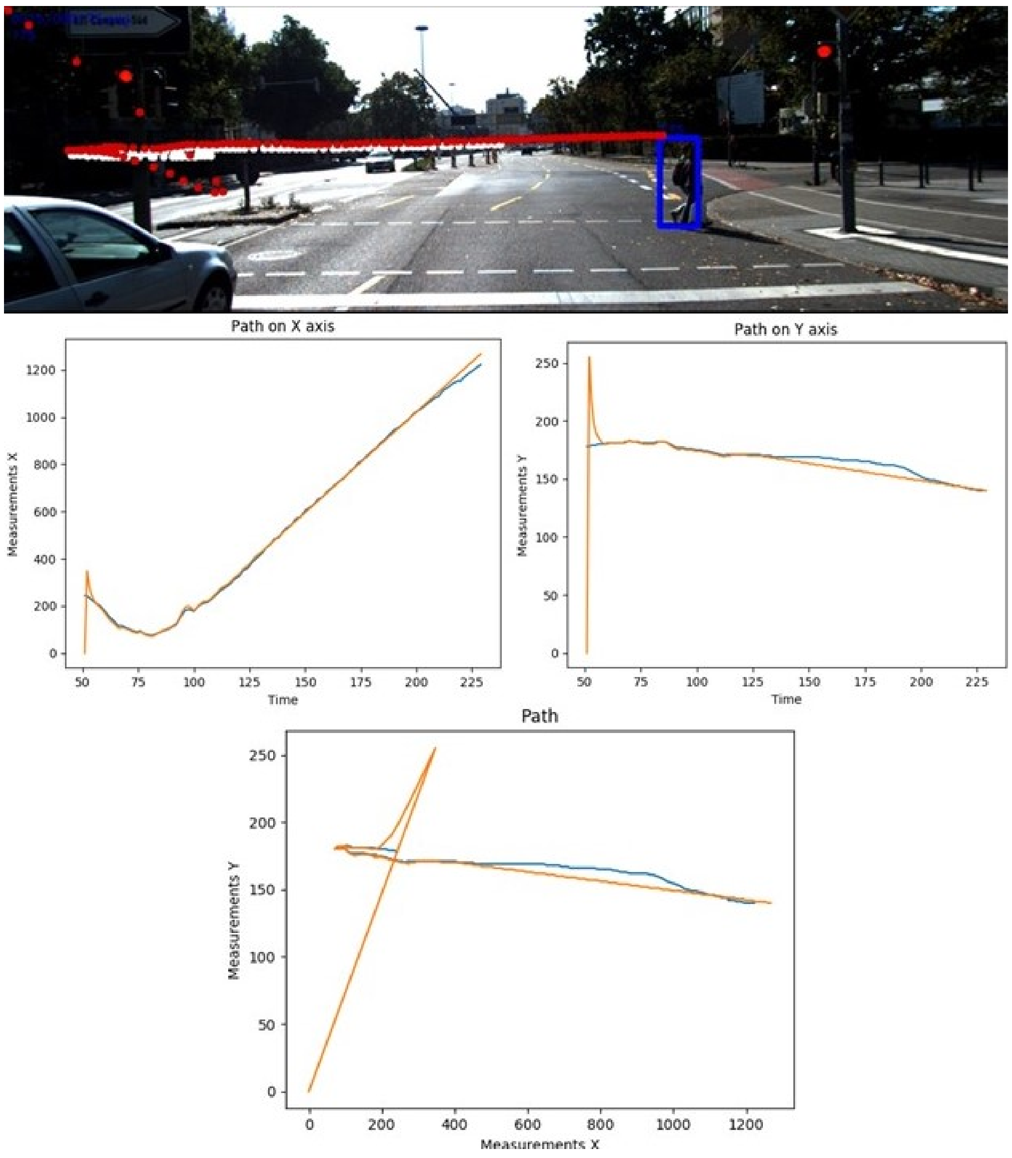


Figure 2.8: The Kalman filter's behaviour. Top: Path prediction on image coordinates. White: actual path. Red: predicted path. Middle-left): Path of the object on the x axis. Middle-right: Path of the object on the image y axis. Bottom: Prediction on x, y axis. Blue: Measurements. Orange: Kalman filter prediction and update.

In other words, the Kalman filter is an algorithm that uses a series of data observed over time, which contains noise and other inaccuracies, to estimates future measurements with more accuracy [50]. Because of its efficiency the Kalman filter has been used in several application such as navigation [11, 59], object tracking [22, 32] and of course path prediction [33, 40]. Because of its wide application and availability of information to understand its algorithm, the Kalman filter is used as a baseline in this research. However due to its limitation to process different type of information its exploration led us to look for more complex techniques, specifically Long Short-Term Memory architectures (LSTMs).

2.3.2 RNN and LSTM Architectures

RNN (Recurrent Neural Network) and Long Short-Term Memory (LSTM) architectures are of special interest to this research, since these types of Neural Networks are widely used in sequential data problems where observed sequence of data is processed to predict the next data in the given sequence, so it is important to have a clear understanding of how they work. Because of that, this section describe these two types of networks putting more emphasis on LSTMs, which are leveraged in this research thesis.

Overview

If we are looking at a set of sequences of numbers, i.e. $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$, $[0, 2, 4, 6, 8, 10, 12]$, $[0, 3, 6, 9, 12]$ and we want to say what is likely to come next, this is a prediction problem where for a given number the goal is to predict the next number in the sequence. For example here, if we only see the number 12 it is difficult to determine what is next, but given some context, looking back in the sequence, the next number becomes easier and easier to predict as the sequence is analyzed backwards. Recurrent neural networks (RNNs) and Long Short-Term Memory Networks (LSTMs) are designed for applications where the input is an ordered sequence and where data from earlier in the sequence may be important.

RNNs are networks that reuse the output from the previous step as an input for the next step. Like a normal neural network the nodes perform a calculation using the input and returns an output value. In an RNN, this output is used along with the next element in the sequence as input for the next step and so on.

LSTMs are a type of RNNs that have recurrent nodes but they also have an internal state. The nodes use this internal state as a working memory space. Which means that the information can be stored and retrieved over many time steps. In an LSTM, 1) the input value, 2) the previous output and 3) the internal state are all used in the node's calculations. The results of the calculation are used not only to provide an output value but also to update the internal state. Like any neural network, LSTMs nodes have parameters that determine how the inputs are used in the calculation. But LSTMs have parameters known as gates that control the flow of information within the node, in particular, how much the saved state information is used as an input to the calculations. The gate parameters are weights and biases, which mean their behaviour depends on the inputs. For example, an input of 11 does not need much past information as the next number is certainly a 12, but an input of 6 might need to recall greater examples of past information.

Similarly there are gates to control how much of the current information is saved to the state and gates that control how much the output is determined by the current calculation versus the saved calculation. Hence, LSTMs' nodes are certainly more complex than the regular Recurrent nodes but this makes them better at learning the inter-dependencies in sequences of data. This previous description was adapted from [95].

Recurrent neural networks (RNNs)

Recurrent neural networks are feedforward neural networks augmented by the inclusion of edges that span adjacent time steps, introducing a notion of time to the model. At time t, nodes with recurrent edges receive input from the current data point x_t and also from hidden node values h_{t-1} in the network's previous state. The output y_t at each time t is calculated given the hidden node values h_t. Input x_{t-1} at time $t-1$ can influence the output y_t at time t and later by way of the recurrent connections [51].

RNNs have been widely used in many sequence-based prediction tasks such as: handwriting imitation, human gait analysis, human-human interactions. Unlike traditional multilayer perceptron (MLP) neural networks, traditional RNNs have a feedback loop connection that can efficiently capture the temporal dependency in their input time series sequences by maintaining an internal state, called "hidden unit", as shown in Figure 2.9 and Figure 2.10. However, traditional RNNs have difficulties in giving an accurate prediction when it comes to memorizing previous lengthy sequences. Thus, the Long Short-term Memory (LSTM) RNN architecture was introduced to help address this problem of traditional RNNs [79].

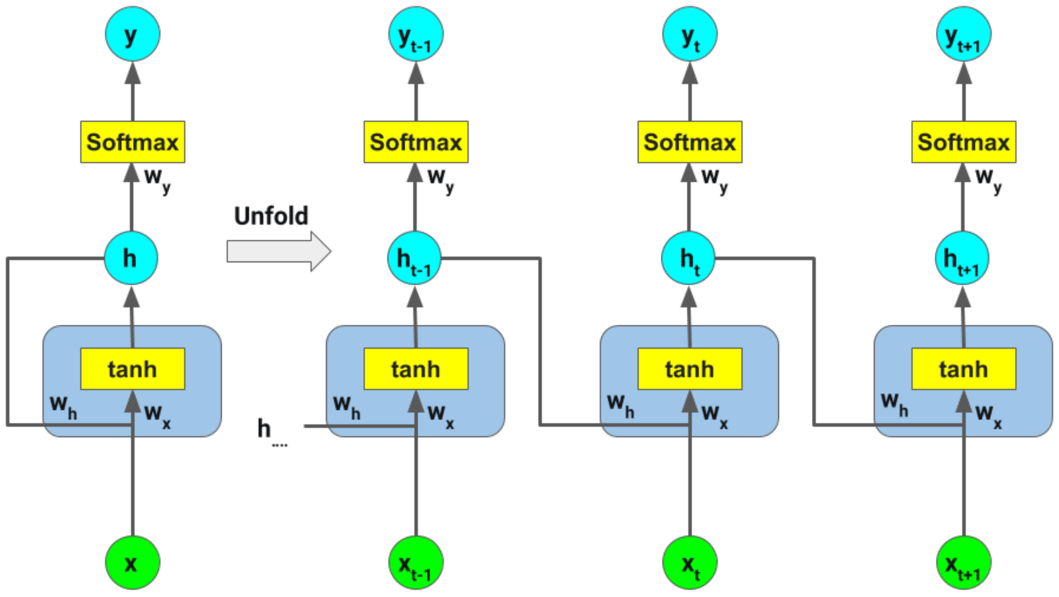


Figure 2.9: Folded and unfolded representations of RNNs [121].

The simple RNN illustrated in Figure 2.10 can be expressed as follows:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h) \tag{2.3}$$

$$y_t = softmax(W_y h_t + b_y) \tag{2.4}$$

Where x_t is the current input vector, h_{t-1} is the previous hidden state, h_t is

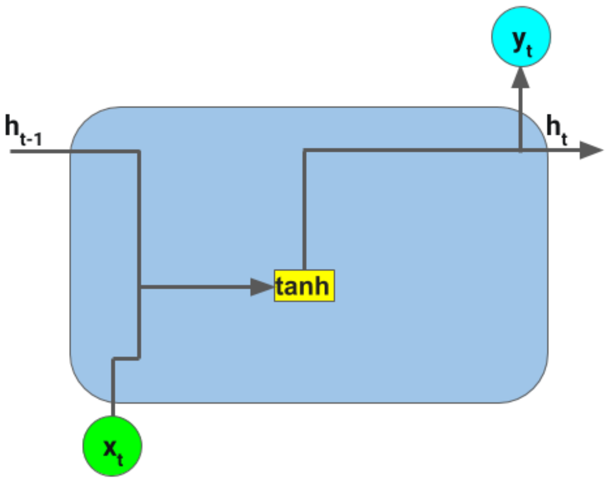


Figure 2.10: Core of a RNN unit.

the current hidden state. W_x, W_h, W_y, b_h, b_y are the weight matrices and variable biases. σ is the activation function normally a $tanh$ function. y_t is the output vector, which is h_t passed through a $softmax$. [51, 121].

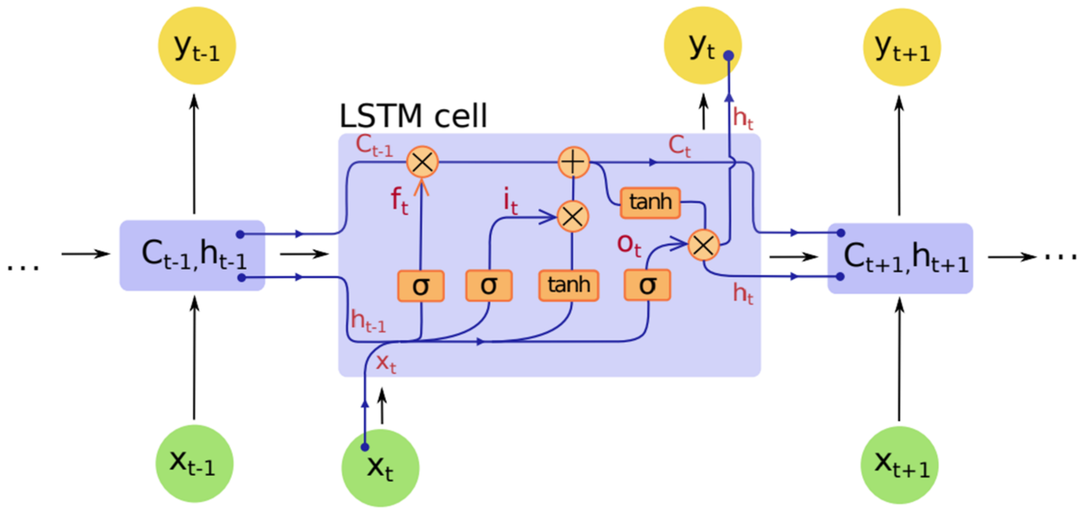


Figure 2.11: Long Short-Term Memory network and LSTM unit [121].

Long Short-Term Memory networks (LSTMs)

While an RNN has internal memory to process sequence data, it suffers from gradient vanishing and exploding problems when processing long sequences. LSTMs (Long Short-Term Memory networks) were specifically developed to address this limitation. Introduced first by [3], several variants were proposed. The “forget gate” for example, that was not included in the original architecture was proposed by [5], and because its use improved the performance of the original LSTMs this gate is standard in most modern implementations.

LSTM Key Concepts

The term “long short-term memory” comes from the following intuition. Simple recurrent neural networks have long-term memory in the form of weights. The weights change slowly during training, encoding general knowledge about the data. They also have short-term memory in the form of ephemeral activations, which pass from each node to successive nodes. The LSTM model introduces an intermediate type of storage via the memory cell. A memory cell is a composite unit, built from simpler nodes in a specific connectivity pattern, with the novel inclusion of multiplicative nodes [51].

The main component of LSTMs is the cell state, which is the horizontal line running through the top of the diagram in Figure 2.11. The cell state transports information and it is responsible for carrying long-term dependencies or patterns. It goes straight through the entire LSTM unit, with only some minor linear interactions. It is by means of these interactions that an LSTM removes or adds information to the cell state. The interactions are performed by an internal mechanism called “gates”. As mentioned by [79], each gate is a composition of a sigmoid neural network layer and an element-wise multiplication operation. The sigmoid layer converts its input into a value between 1 and 0 – in this way having the role of a gate. These gates regulate the flow of information and are responsible for learning which information in the sequence is relevant and should be kept. 1 means that the gate is fully open, so information flows through while 0 indicates that the gate is closed so information will be blocked.

LSTM architectures

LSTMs employ three gates, including a forget gate, input gate, and output gate, to modulate the information flow across the cells and prevent gradient vanishing and explosion [121], as presented in Figure 2.11. The forget gate is responsible for deciding which information from the current input and the previous cell state flows into the current cell state. The input gate is responsible for deciding which values from current input update the current cell state. The output gate is responsible for deciding what to output as the next hidden state based on the current input and the current cell state [79], which is formulated as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.5}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.6}$$

$$\tilde{\mathbf{C}}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{\mathbf{C}}_t \tag{2.8}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.9}$$

$$h_t = o_t * tanh(C_t) \tag{2.10}$$

where f_t, i_t, o_t, $\tilde{\mathbf{C}}_t$ and c_t are the activations for the forget, input, output, cell state candidate and cell state gates at time t respectively. While W_f, W_i, W_o, W_c, b_f, b_i, b_o, b_c are their respective weight matrices and variable biases. x_t is the memory cell input and h_t is the next hidden state and final output at time t. σ represents the activation function sigmoid, and $tanh$ is the activation function tanh.

Intuitively, in terms of the forward pass, the LSTM can learn when to let activation into the internal state. As long as the input gate takes value zero, no activation can get in. Similarly, the output gate learns when to let the value out. When both gates are closed, the activation is trapped in the memory cell, neither growing nor shrinking, nor affecting the output at intermediate time steps. In terms of the backwards pass, the constant error carousel enables the gradient to propagate back across many time steps, neither exploding nor vanishing. In this sense, the gates are learning when to let error in, and when to let it out. In practice, the LSTM has shown a superior ability to learn long-range dependencies as compared to simple RNNs [51]. A detailed description of RNNs and LSTMs is given by [51, 119].

LSTMs step by step

LSTMs process information in four steps that can be termed:

1. Forget: discards irrelevant historical information,
2. Store: keeps relevant parts of new information,
3. Update: using information of the steps one and two, selectively update the cell state.
4. Output: generate an output.

The first step in an LSTM, as shown in Figure 2.12, is to decide what information is going to be thrown away from the previous cell state C_{t-1}. This decision is made by the forget gate f_t. To do that, f_t looks at the previous hidden state h_{t-1} and information of the current input x_t. f_t outputs a number between 0 and 1 for each number in the cell state C_{t-1}. Remember that a 0 means to forget or throw away and a 1 means to remember or to keep such information.

The second step, depicted in Figure 2.13, is to decide what new information is going to be stored in the cell state. This has two parts, first the input gate,i_t, decides which values are going to be updated by looking at h_{t-1} and x_t. Next, the tanh function creates a vector of new candidate values, $\tilde{\mathbf{C}}_t$ that could be added to the new cell state C_t, by processing the same h_{t-1} and x_t. In the next step, these two will be combined to create an update to the state.

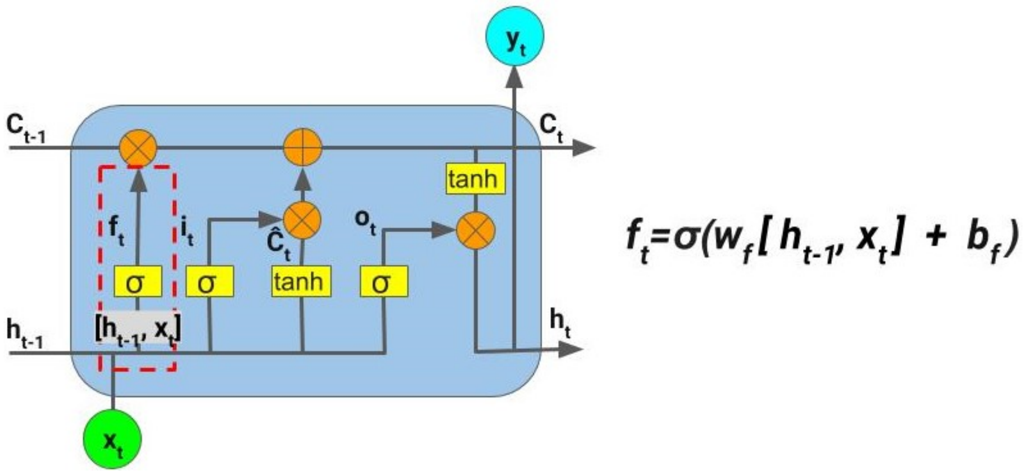


Figure 2.12: Long Short-Term Memory network process: step one.

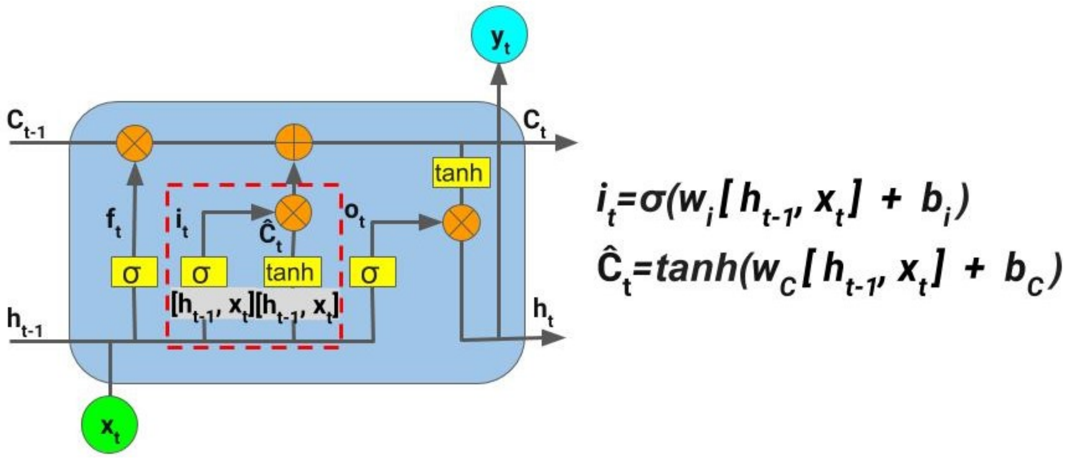

y_t
C_{t-1}
C_t
tanh
f_t
i_t
$\hat{C}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
$[h_{t-1}, x_t][h_{t-1}, x_t]$
h_t
x_t
$i_t = \sigma(w_i [h_{t-1}, x_t] + b_i)$
$\hat{C}_t = tanh(w_C [h_{t-1}, x_t] + b_C)$

Figure 2.13: Long Short-Term Memory network process: step two.

At this point, the third step has enough information to calculate the new cell state, C_t. This next step, illustrated in Figure 2.14, is to update the previous cell state, C_{t-1}, into the new cell state C_t. First, the previous cell state, C_{t-1}, is multiplied by the forget gate, f_t, forgetting the things already decided in the first step. Then, by a point-wise addition, $i_t \times \tilde{\mathbf{C}}_t$ is added to that. Remember, $i_t \times \tilde{\mathbf{C}}_t$ are the new candidate values, scaled by how much it was decided to update each state value. This step gives the new cell state, C_t.

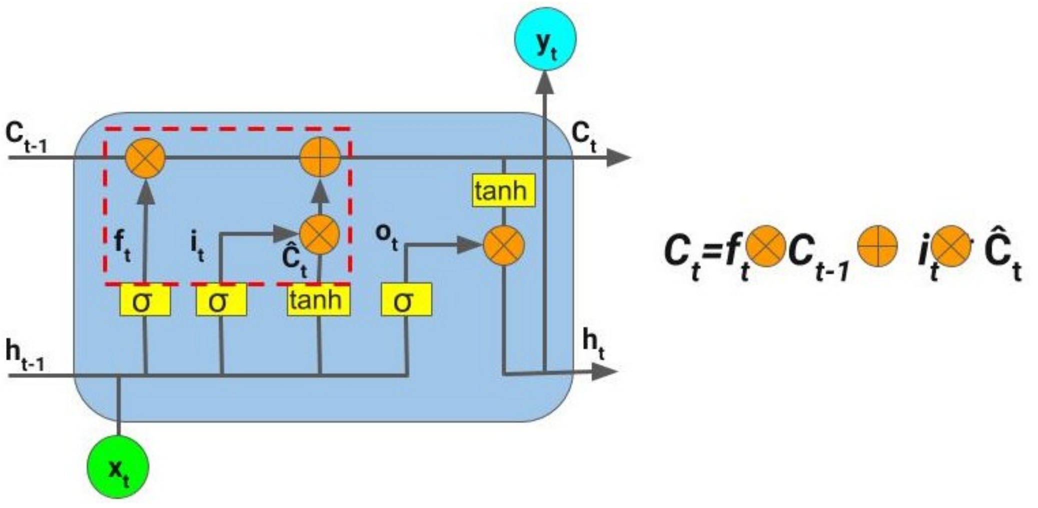


Figure 2.14: Long Short-Term Memory network process: step three.

Finally step four, shown in Figure 2.14, has to decide what is going to be the next hidden state, h_t, and output, y_t. This is based on the new cell state C_t, but is multiplied by the output gate o_t. o_t decides which part of the new cell state is going to be the output. First the previous hidden state and current input information,$C_{t-1} + x_t$, are passed to the input gate, o_t. Next, the cell state goes through tanh (to push the values to be between -1 and 1). Then o_t and the tanh output are multiplied to decide which information the next hidden state should carry and what information is output at that time step. Finally, the next hidden state, h_t, and the new cell state, C_t, are passed to the new time step.

Some important characteristics to remember about LSTMs are that they:

- Maintain a separate cell state from what is output
- Use gates to control the flow of information.
  - Forget gate discards irrelevant information from past in the sequence.
  - Store gate saves important information from the input at the current time step.
  - Update gate selectively calculates the new cell state.

  - Output gate gives the new hidden state that goes to the next time step and also say what information is actually output at that time step.

- Perform back-propagation through time with uninterrupted gradient flow by modulating the information flow across the cells and prevent gradient vanishing and explosion.

This previous description was adapted from [125, 124].

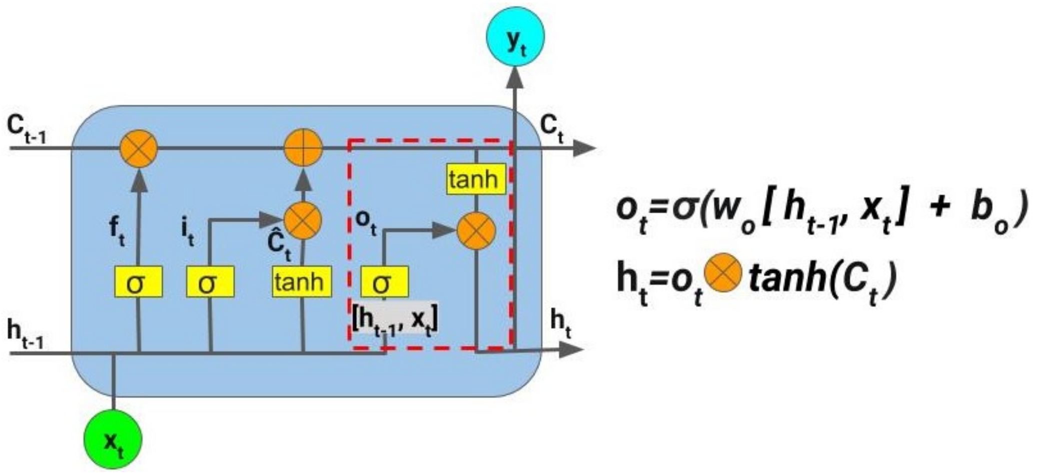


Figure 2.15: Long Short-Term Memory network process: step four.

LSTM architectures are currently used in areas such as language translation [45, 51], time series prediction [64, 76] and trajectory prediction [85, 57, 66, 84, 74]. Becase LSTMs are capable of getting information from sequences and then predicting using that previous information.

To predict the future path of an object in a scene, we have to analyse the previous information of such object, i.e. previous x, y positions (tracklets) or any available information around the object as shown in 2.16. Instrumented vehicles provide a rich set of information that can be leveraged for path prediction. An important characteristic is that this information is given sequentially in an ordered manner, this means that the information is time dependant and as such past information can be leveraged to predict future information (in this case future path of objects). Due to the characteristics of path prediction data, LSTM architectures are suitable to tackle this problem, since this type of networks are able to process sequential data and leverage information through time, and based on that perform a better prediction.

As described in this subsection, LSTM architectures are capable of leveraging past information in a sequence to predict future values, i.e observed past path to predict a future path. However, LSTMs have the limitation of only able of predicting a single path per observed tracklet. Regarding that, the next subsection describes how Mixture Density Network can be leveraged along with LSTMs to be able to predict multiple paths.

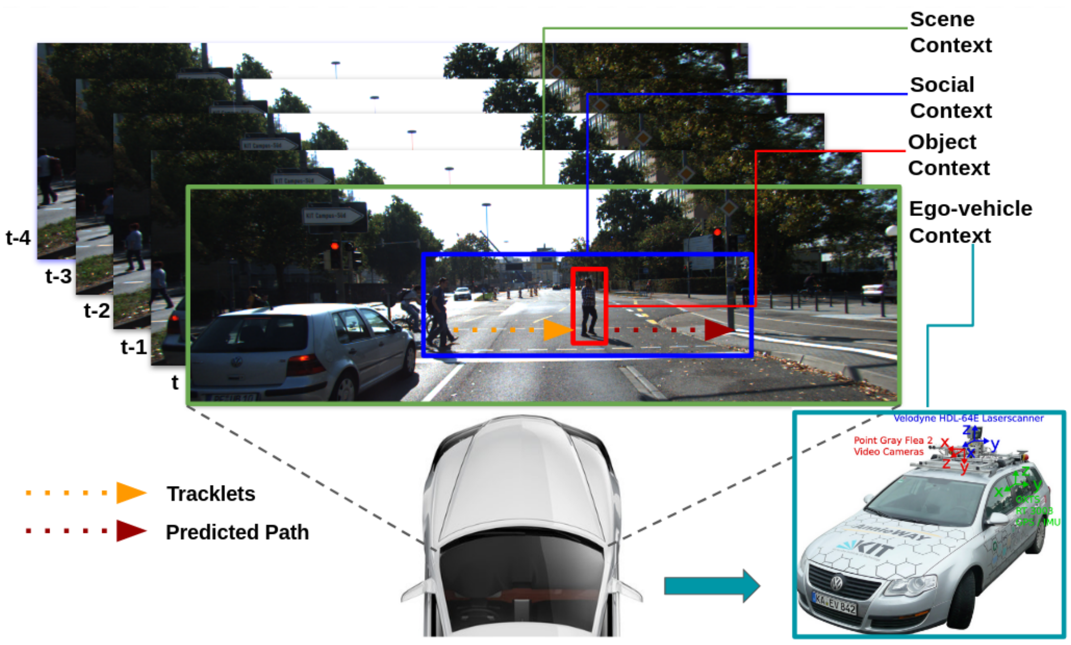


Figure 2.16: Contextual information from Ego vehicle view

2.3.3 Mixture Density Networks

LSTM architectures can only predict a single path per observed tracklet, fortunately work done on Mixture Density Networks (MDN) can be leveraged to overcome this limitation.

A Mixture Density Network is a type of network introduced by [2] where that replaces the Gaussian distribution with a mixture model with the flexibility of modeling completely general distribution functions. In this way, the probability density of the target data is represented as a linear combination of kernel functions, as illustrated in Figure 2.17.

Since then, Mixture Density Networks (MDNs) have been applied on different works such as modeling of handwriting [27] or for generation of sketch drawings [71]. At this point MDNs were not combined with standard Neural Networks, such as a multi-layer perception, but with more complex architectures such as Recurrent Neural Network (RNN), more specifically with LSTMs. In this way, the LSTM gives memory to the input network and the mixture model contributes uncertainty to the output. An interesting work is described in [104] where they generate images based on a sequence of past observed images using LSTMs and MDNs and also make a study on the role of the different mixture components.

A highly related work is presented in [114], where the authors predict the intention of the driver (left, straight, right, u-turn) at 5 determined intersections in a static birds-eye view by predicting the future trajectories. In the intersection the possible trajectories of a vehicle are constrained to the five scenes and they apply clustering to the set of trajectories on the dataset to filter the predictions. In [90] the prediction is using static cameras, from synthetic test conditions and some real scenarios, but instead of predicting the intention of the driver they predict the trajectory of pedestrians.

2.3.4 Computer Vision and CNNs

While LSTMs and MDNs can work with only the x, y positional information of objects, this research thesis is looking to leverage more data available from instrumented vehicles. Cameras are one of the common used sensors which provides images of the surrounding of a vehicle, such information is aimed to be used to give more context in the path prediction task.

Computer vision has become increasingly important and effective in recent years due to its wide-ranging applications in areas as diverse as smart surveillance and

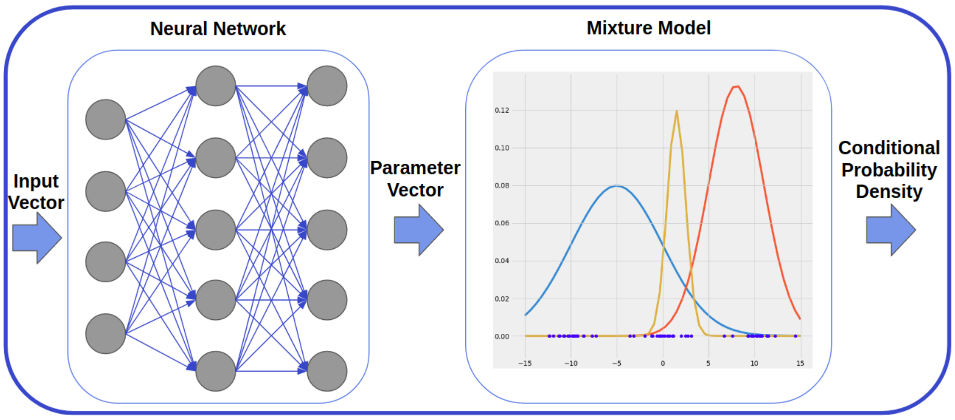


Figure 2.17: MDNs consist of a feed-forward neural network whose outputs determine the parameters in a mixture density model. The mixture model then represents the conditional probability density function of the target variables, conditioned on the input vector to the neural network [2].

monitoring, health and medicine, sports and recreation, robotics, drones, and self-driving cars. Visual recognition tasks, such as image classification, localization, and detection, see Figure 2.18, are the core building blocks of many of these applications, and recent developments in Convolutional Neural Networks (CNNs). Introduced first by [1], have led to outstanding performance in the state-of-the-art of visual recognition tasks and systems. As a result, CNNs now form the basis of deep learning algorithms in computer vision [93]. [44] also evidences how CNNs have shown good performance on feature extraction for several task such as image classification, fine grained recognition and attribute detection.

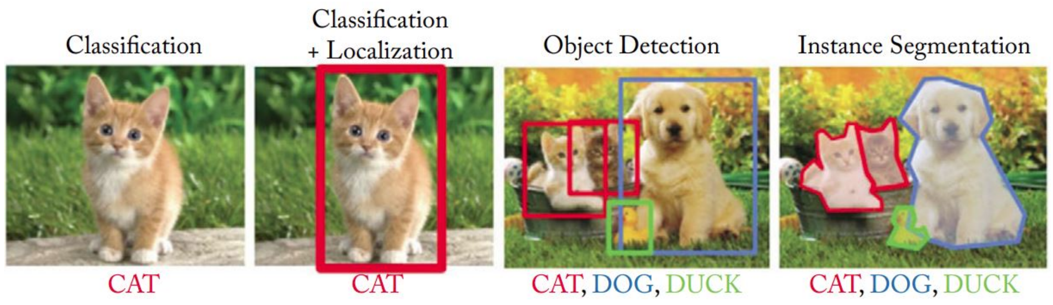


Figure 2.18: What do we want computers to do with the image data? To look at the image and perform classification, classification plus localization (i.e., to find a bounding box around the main object (CAT) in the image and label it), to localize all objects that are present in the image (CAT, DOG, DUCK) and to label them, or perform semantic instance segmentation, i.e., the segmentation of the individual objects within a scene, even if they are of the same type [93].

A convolutional neural network usually stacks a sequence of convolutional (Conv)-ReLU layers, followed by the pooling layers (Pool), and repeats this pattern until the image has been merged spatially to a small size. At some point, it is common to transit to fully-connected layers (FC). For clarity, the most common CNN architecture follows the pattern: Input $\Rightarrow$ [[Conv $\Rightarrow$ ReLU] $*$ n $\Rightarrow$ Pool?] $*$ m $\Rightarrow$ [FC $\Rightarrow$ ReLU] $*$ l $\Rightarrow$ FC, where the $*$ indicates repetition, and the question mark ? indicates an optional layer. In addition $n \geq 0$ (and usually n $\leq$ 3), m $\geq$ 0, l $\geq$ 0 (and usually l $<$ 3). We can take the responses from one of the network's layers as our CNN feature vector, which can be used for different visual tasks in combination with some other techniques [107]. [46] presents a work on the visualization of the learned features of CNN layers and [93] gives a good summary of that in Figure 2.19.

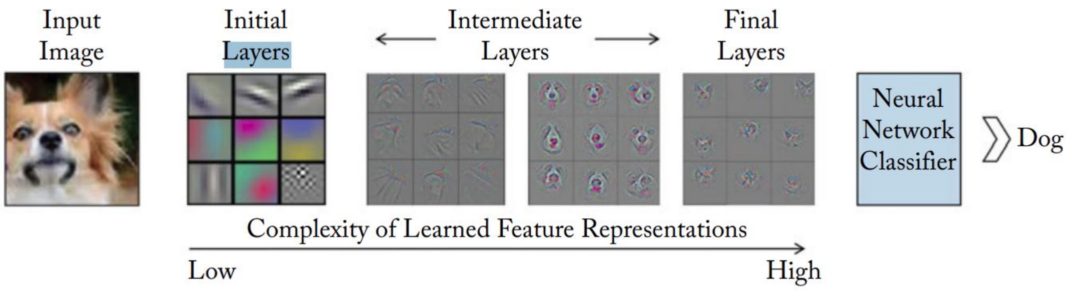

Input Image
Initial Layers
Intermediate Layers
Final Layers
Neural Network Classifier
Dog
Complexity of Learned Feature Representations
Low
High

Figure 2.19: A CNN learns low-level features in the initial layers, followed by more complex intermediate and high-level feature representations which are used for a classification task [93].

2.4 Literature Review on Path Prediction

In the context of autonomous vehicles, reaching the point of being able to predict the motion of objects in the scene has several previous tasks that have to be solved first. Tasks such as sensor data synchronization, understanding/sensing of the surroundings (object detection and classification, tracking, scene segmentation). Visualizing this task into modules, [75] presents a diagram with the modules that would constitute an autonomous vehicle, Figure 2.20. According with [75], the Traffic Prediction module is responsible for predicting the future behavior of surrounding objects such as pedestrians, vehicles and, cyclists identified by the Perception module. It outputs predicted behaviours that are fed into downstream planning and control modules as data input.

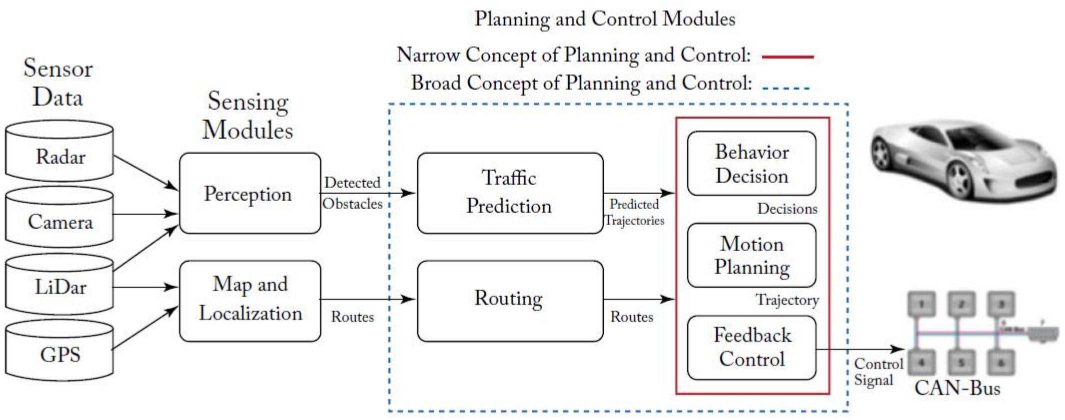


Figure 2.20: Planning and control modules under narrow and broad concepts [75].

Depending on the output information that has to be predicted, prediction can be seen as a classification or as a regression problem [66].

- Classification of object's behaviour: the aim is to predict the behaviour of the observed object such as stopping, going straight, turn left, turn right, etc.
- Regression for path prediction: the information to be predicted is the future position of the observed object in a given space at at time t, $tr^P = [tr^P_{t1}, tr^P_{t2}, ..., tr^P_{tpred}]$

Both type of information given by the prediction module are important depending on the downstream task where this is needed.

This research thesis is focused on the problem of prediction as a regression problem since the aim of this research work is to predict the path of the objects around the ego vehicle.

Path prediction has been addressed using different approaches according to the type of information that is taken into account. The purpose of this chapter is to provide a review of the existing works that have been published in this research area to identify techniques suitable for further investigation. This literature review is structured according the approach used: physical-based, manoeuvre-based and interaction-aware. This classification was adopted from [41], who presented a survey on motion prediction and risk assessment for vehicles. In this literature review this classification is adopted and applied to path prediction in general. [41] also mention the limitation of each approach, such limitations are interesting since they have to be taken into account in this research thesis.

2.4.1 Physical-based

Physical-based approaches take into account only the object as an isolated identity and consider the laws of physics that govern it. There are some works like [6] that use information from the ego-object, i.e: wheel steering angle, vehicle speed for predicting the path (dynamic models), but that information is hard to obtain if we want to predict the path of the other objects in the scene not only the ego vehicle.

The simplest approach that is widely used is the Kalman Filter (KF) along with kinematic models and several variants such as the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) [91]. [33] presents a comparative study of the Kalman filter with some kinematic models in a vehicle context. They studied several single kinematic models such as Constant Velocity (CV), Constant Acceleration (CA), and Constant Turn Rate (CT) and interacting multiple models (IMMs) and the results show no significant performance gain of the more sophisticated IMMs considered vs. the simpler CV, for current position estimation. They attribute that to the high sampling rate and the low measurement error. Another example of using the KF is found in [12] where they use the Extended KF to perform short term prediction.

Other related work is shown in [40], in this research four different methods were evaluated for predicting future pedestrian positions accurately: Gaussian process dynamical models (GPDMs) and probabilistic hierarchical trajectory matching (PHTM) that use augmented features derived from dense optical flow and KF and IMM that use positional information only. In this work, they conclude that similar path prediction performance was reached for the four approaches on walking motion, with near-linear dynamics. During stopping, however, the newly proposed approaches (SFlowX/GPDM or HoM/Traj), with nonlinear and/or higher order models and augmented motion features, achieved a more accurate (longitudinal and lateral) position prediction of 10–50 cm at a time horizon of 0–0.77 seconds.

In [52] they predict the position of a pedestrian using a particle filter. This work is interesting to mention because it makes use of a map to warn the driver when a pedestrian is in a hazardous area. An observation here is that they do not use the map to constrain the possible movement of a pedestrian. [85] present interesting work on prediction of pedestrian trajectories from a camera mounted in a vehicle where they use a Bayesian Recurrent Neural Network as shown in Figure 2.21. They predict the future bounding boxes by taking into account the past bounding box sequence, the past odometry sequence and the future corresponding odometry sequence. The future odometry sequence is predicted according to the past odometry sequence and visual information. They compare their results with Kalman Filter, LSTM, LSTM-Aleatoric, LSTM-Bayesian and LSTM-Bayesian with odometry information (proposed approach). Experimental results showed that their approach outperforms all the other methods.

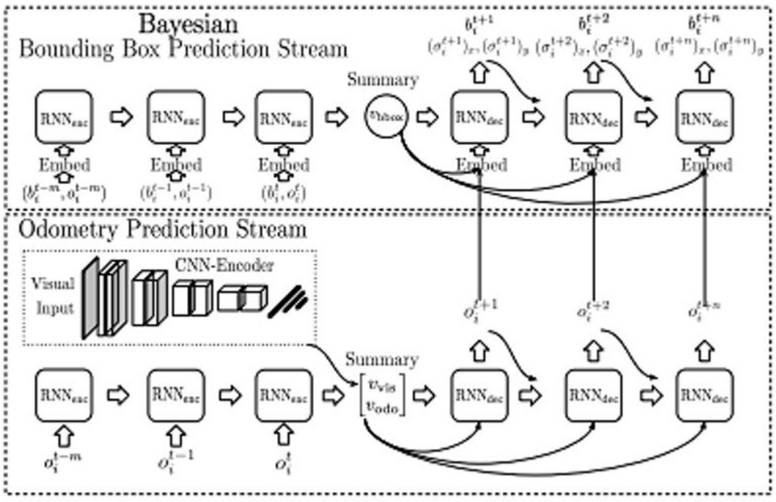


Figure 2.21: Two stream architectures for prediction of future pedestrian bounding boxes [85].

Another related work is presented in [57], where they address the problem of predicting the trajectory of pedestrians in crowded spaces using static cameras. This approach, called Social LSTM, uses one LSTM for each of the pedestrians in the scene. Social refers to the use of the trajectory of other pedestrians that is taken into account to predict the trajectory of a single one. They use a separate LSTM for each trajectory and then connect each LSTM to other through a Social pooling layer, this pooling layer allows spatially proximal LSTMs to share information. The hidden states of all LSTMs within a certain radius are pooled together and used as an input at the next time step. Similar work is presented in [66] where they use LSTMs to predict the trajectory of vehicles in highways from a fixed top view perspective. In [84] multiple cameras were used to predict the trajectory of people in crowded scenes and [74] predict the trajectory of vehicles in an occupancy grid from the perspective of ego vehicle. [85, 57, 66, 84, 74] show a trend in using LSTM architectures to process past observed positions of objects to predicted their future trajectory. So they treat a path as a sequence of data (time series) and the LSTMs, which are neural network able to process sequential data taking into account long and short dependencies in the sequence, help to predict the next n future positions of objects.

Limitations

Since they only rely on the low level properties of motion (dynamic and kinematic properties), Physics-based motion models are limited to short-term (less than a second) motion prediction. Typically, they are unable to anticipate any change in the motion of the car caused by the execution of a particular manoeuvre (e.g. slow down, turn at constant speed, then accelerate to make a turn at an intersection), or changes caused by external factors (e.g. slowing down because of a vehicle in front) [41].

2.4.2 Manoeuvre-based

These kind of approaches can be based on prototype trajectories or based on manoeuvre intention estimation. In [8] they perform motion prediction. The intuition they take as starting point is that "for a given area, moving objects tend to follow typical motion patterns that depend on the objects' nature and the structure of the environment". In this work they use the Expectation-Maximization learning algorithm to cluster all the trajectories from a specific scenario and then using these clusters predict the motion for a partially observed trajectory. Since cluster-based techniques permit to take into account not only the current state of the object but also its past states, they are the preferred approaches when it comes to long term motion prediction. Their weakness lies in their inability to predict atypical trajectories and also that they are designed for specific scenarios. A similar work is found in [63] and [16] where they also use clustering techniques and hidden Markov models (HMM) respectively. A highly interesting survey about trajectory clustering can be found in [86].

In [12] the authors address long-term prediction by classifying two actions of the vehicle: 1) speed profile: Quick Acceleration (QA), Slow Acceleration (SA), Keep the same Speed (KS), Quick Deceleration (QD), Slow Deceleration (SD) and 2) changing of lane: staying in its lane, going into the right lane, going into the left lane. For each vehicle on the road, they compute all possible sequences of action, regarding the current velocity and location. They assign a cost (costs in this context are determined using a heuristic approach) for each action and unrealistic action sequences are eliminated. Their algorithm uses the sequences with the highest probability to determine the future location of vehicles. This work is interesting because they fuse the results of a short-term and a long-term algorithm to strengthen or weaken the prediction of the other. A similar approach is found in [29] but in contrast this work predicts the trajectory of the ego vehicle. In [40] the authors, classify the action of a pedestrian in the curbside (stopping or walking). They evaluate four techniques involving Gaussian process dynamical models (GPDMs), probabilistic hierarchical trajectory matching (PHTM), Kalman filter (KFs), and its extension interacting multiple model KF (IMM KF). To provide some context, they also evaluate human performance for the same test. From the experiment results they noticed, "The humans reach accuracy of 0.8 in classifying the correct pedestrian action about 570 ms before the event. This accuracy is only reached about 230 ms before the event by the newly developed SFlowX/GPDM and HoM/Traj systems, which use augmented visual features. The baseline IMM-KF system does worst, reaching the corresponding accuracy only about 90 ms before the event". A similar work is reported in [31] where they detect the pedestrian's intention to enter the traffic lane at intersections.

The previous works are worthwhile mentioning. However for the purpose of this research they have two main constraints. The first one is that clustering techniques cannot predict atypical trajectories and they are created for predicting trajectories in specific scenarios with static background. The second limitation is that for using approaches that first classify the actions that the object is performing labeled data of those actions is needed. Looking at the available datasets in the context of an ego vehicle in traffic scenes, which is the main focus of this research, they normally present dynamic back ground since the cameras are mounted on a moving vehicle and also these datasets do not provide labeled data of the action of the objects.

Limitations

In practice, the assumption that objects move independently from each other does not hold. Objects share the scene with other objects, and the manoeuvres performed by one object will necessarily influence the manoeuvres of the other objects. Disregarding these dependencies can lead to erroneous interpretations of the situations [41].

2.4.3 Interaction-aware

Interaction-aware approaches take into account not only the object as an isolated element but also its surroundings [72, 80]. Of course taking into account all elements in the scene is complex and computationally expensive so, some works only take into account certain elements such as the type of object, the location, and the static surroundings.

One related work is shown in [62] where they predict the path of an object based on a reward map. This reward map is a segmented image of the scene with the regions that a vehicle or a pedestrian is more like to move through, i.e a pedestrian is more likely to walk on the side walk than on the road. They transform the problem of path prediction into a directed graph problem which has to be optimized based on a reward map. [62] use deep learning to create this reward map and another similar work [69] address this using HOG. Even though both articles work on static cameras and 2D images, this idea can be used in a vehicle context by using semantic segmentation. In [52] for instance, they use information of a map and GPS to see where the pedestrians are located in the scene, but they only use this information to warn the driver if the pedestrian is in a dangerous area. Semantic segmentation is showing promising results that can be leveraged in this research [18, 26, 60] and in this way replace the use of a map and a GPS. Semantic segmentation can also be used to get local information such as dynamic objects that cannot be captured by a map or GPS. [99] present work on future person localization, but this time they work with videos from a camera carried by a person. In order to predict future position of a person they take into account, besides the location of the person, also the scale of the bounding box, the pose and the ego-motion features of the ego vehicle and propose a network architecture to fuse such information as shown in Figure 2.22 and Figure 2.23.

As can be noted, the mentioned works present a trend in using other features from the scene besides only using x, y positional information of the objects to predict their future path. This motivates the idea of developing a holistic path prediction approach able to process more features from the traffic scenes along with the x, y positional information of the objects leading to improve performance in the path prediction task.

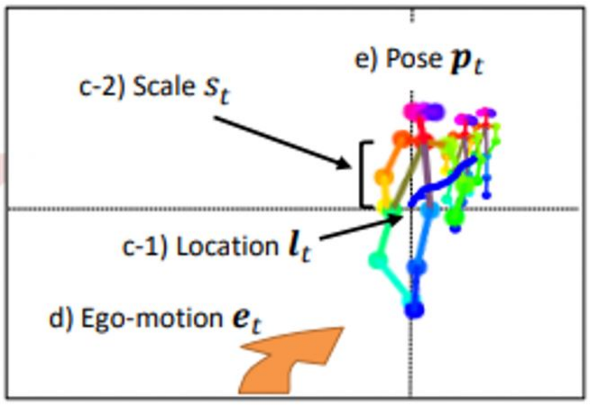


Figure 2.22: Information taken into account in the prediction of future position [99].

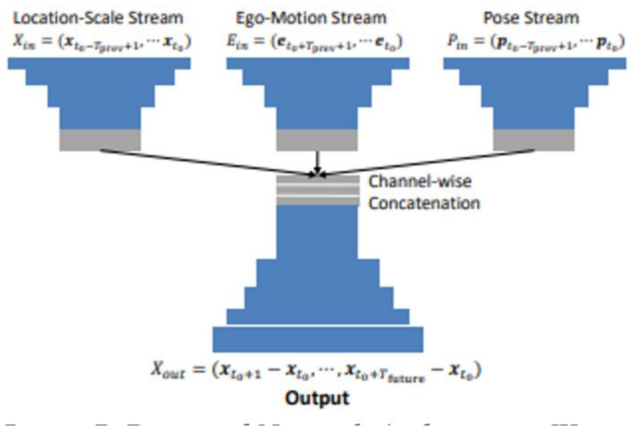

Location-Scale Stream
$X_{in} = (\boldsymbol{x}_{t_0-T_{prev}+1}, \cdots \boldsymbol{x}_{t_0})$
Ego-Motion Stream
$E_{in} = (\boldsymbol{e}_{t_0+T_{prev}+1}, \cdots \boldsymbol{e}_{t_0})$
Pose Stream
$P_{in} = (\boldsymbol{p}_{t_0-T_{prev}+1}, \cdots \boldsymbol{p}_{t_0})$
Channel-wise Concatenation
$X_{out} = (\boldsymbol{x}_{t_0+1} - \boldsymbol{x}_{t_0}, \cdots, \boldsymbol{x}_{t_0+T_{future}} - \boldsymbol{x}_{t_0})$
Output

Figure 2.23: Proposed Network Architecture [99].

Limitations

The Interaction-aware motion models are the most comprehensive models proposed so far in the literature. They allow longer-term predictions compared to Physics-based motion models, and are more reliable than Manoeuvre-based motion models since they account for the dependencies between the vehicles. However, this exhaustiveness has some drawbacks: computing all the potential trajectories of the vehicles with these models is computationally expensive and not compatible with real-time risk assessment [41].

2.4.4 Summary of Path Prediction Approaches

From this literature review, we observed that significant research has been conducted on path prediction in general but work specifically in the context of ADAS systems still poses a challenge. Several of the works mentioned use static cameras, however, these techniques can be adapted for cameras with ego motion. Approaches ranging from Kalman filters, clustering techniques to deep neural networks have been identified. This literature review also identifies a trend that the techniques are following to predict the path of an object where increasing varieties of data, besides x, y positional information of the objects, are taken into account when performing path prediction. It can also be noted that current works tend to fuse different kind of information through deep neural networks. This supports one of the central principle of the research hypothesis: that a holistic, integrated approach will improve prediction performance.

2.5 Traffic Datasets

As mentioned in [109], there are several datasets related to traffic scenes. CityFlow [113], provides information of vehicles from surveillance cameras in image coordinates. Cityscapes [60] contains 2D semantic, instance-wise, dense pixel annotations for 30 classes. BDD100K [122] provides data for object detection, instance segmentation, driveable area, lane markings. Mapillary [77] is a dataset for Panoptic Segmentation and Object Detection (Instance Segmentation) tasks. The Simulation (NGSIM) dataset [13] has trajectory data for cars, but the scene is limited to highways with similar simple road conditions. KITTI [26] is a dataset for different computer vision tasks such as stereo, optical flow, 2D/3D object detection, and tracking. However, the total time of the dataset with tracklets is about 22 minutes. In addition, there are few intersection between vehicles, pedestrians and cyclists in KITTI, which makes it insufficient for exploring the motion patterns of traffic agents in challenging traffic conditions. Waymo is another interesting dataset, it provides 3D Lidar Labels, 2D Camera Labels, for two challenges such as object detection and tracking for the objects pedestrian, vehicle, cyclists [120]. There are some pedestrian trajectory datasets like ETH (Pellegriniet al.2009) [19], UCY (Lerneret al.2007) [14], etc., but such datasets only focus on human crowds without any vehicles.

However, there are some limitations with the aforementioned datasets:

- Do not provide annotated tracklets of the object [60].
- Provide data of only one type of object [13, 19].
- Provide data in only one perspective which is not from a camera mounted on a vehicle [13, 14, 19].

- These datasets do not provide data specifically for path prediction task.

Very recently, some datasets provide data for prediction challenge. ApolloScape [89] is a large-scale comprehensive dataset of street views that contains higher scene complexities, 2D/3D annotations and pose information, lane markings and video frames. Recently, ApolloScape [109] made available data for trajectory prediction. It contains highly complicated traffic flows mixed with vehicles, riders, and pedestrians. They manually selected scenarios where the interaction among object were complex. Similar, the nuScenes [115] dataset which is inspired by the pioneering KITTI dataset, is the first large-scale dataset to provide data from the entire sensor suite of an autonomous vehicle (6 cameras, 1 LIDAR, 5 RADAR, GPS, IMU). Compared to KITTI, nuScenes includes 7x more object annotations. It provide data for detection and tracking tasks. This year nuScenes started also to provide data for trajectory prediction. Lift [108], another dataset for autonomous vehicles, releases this year data for motion prediction [118]. Argoverse [102], provides data for 3D tracking and motion forecasting.

Dataset selection. Several datasets were explored and analysed, however from them, two datasets were selected, KITTI and CityFlow. KITTI was selected because most of the datasets that provide data for forecasting give only the positions of the objects in world coordinates and provide a map of the scenes. However, these datasets do not provide RGB images of the scenarios and ego vehicle features as KITTI does. KITTI provides information from different type of sensors that can be fused and used together, which is desirable in this research work where we are focused in using different type of information present in a normal traffic scene. Some of the new datasets provide more information of the surrounding of a vehicle. However, they started to provide this information too recently to be included in this research. Besides, this research uses KITTI because it is widely used in the literature of autonomous vehicle and has been available since the beginning of this thesis so we are familiar with the structure of the data.

The other dataset used in this research was CityFlow which provides data from static cameras. This dataset was selected because it provides annotated data of vehicles from surveillance cameras from different traffic scenarios. We were already familiar with the structure of the data on this dataset, so its use did not require any further analysis on understanding the data and its format. A more complete description of KITTI and CityFlow datasets is given below:

KITTI [26]: is one of the most popular datasets for use in mobile robotics and autonomous driving. It has realistic scenarios with a variety of objects such as in the city, highways, crossing road, vehicle standing, moving, etc. It consists of traffic scenarios recorded with a variety of sensor modalities, including high resolution RGB, grayscale stereo cameras, and a 3D laser scanner. Most recently it provides 200 training images as well as 200 test images for semantic segmentation. It also provides 21 sequences with the tracking labels of the objects in image coordinates and 3D information. The resolution of the videos is 1242x375 and are recorded at 10 FPS.

CityFlow [113]: the data used for City-Scale Multi-Camera Vehicle Tracking was obtained. This dataset contains 3.25 hours of videos collected from 40 cameras spanning 10 intersections in a mid-sized U.S. city. The dataset covers a diverse set of location types, including intersections, stretches of roadways, and highways. The dataset is divided into 5 scenarios. Only 3 of the scenarios are used for training, and the remaining 2 are used for testing. The length of the training videos is 58.43 minutes, while testing videos are 136.60 minutes in length. In total, the dataset contains 229,680 bounding boxes for 666 distinct annotated vehicle identities. Only vehicles passing through at least 2 cameras have been annotated. The resolution of each video is at least 1920x1080 and the majority of the videos have a frame rate of 10 FPS. The three scenarios from training were used, since only these scenarios contains the labeled objects with ID (tracks).

2.6 Evaluation Metrics

An important consideration in judging the performance of future path predication algorithms is the evaluation metric. The aim is to describe how closely the predicted path matches that observed or ground-truth path. This section describes how these measures are calculated and some of the issues with using them as a global measure of path accuracy.

Some works report the results in term of the Mean Squared Error (MSE) [67], other in terms of the Root of the Mean Squared Error (RMSE) [87], lately, two main metrics have been used to evaluate the performance of a path prediction approach, Average Displacement Error (ADE) and Final Displacement Error (FDE) [57, 97, 106, 109]. ADE and FDE are defined as follow:

- **Average Displacement Error (ADE):** takes into account the difference between every estimated point of each predicted path and its true path,as described in Figure 2.24.

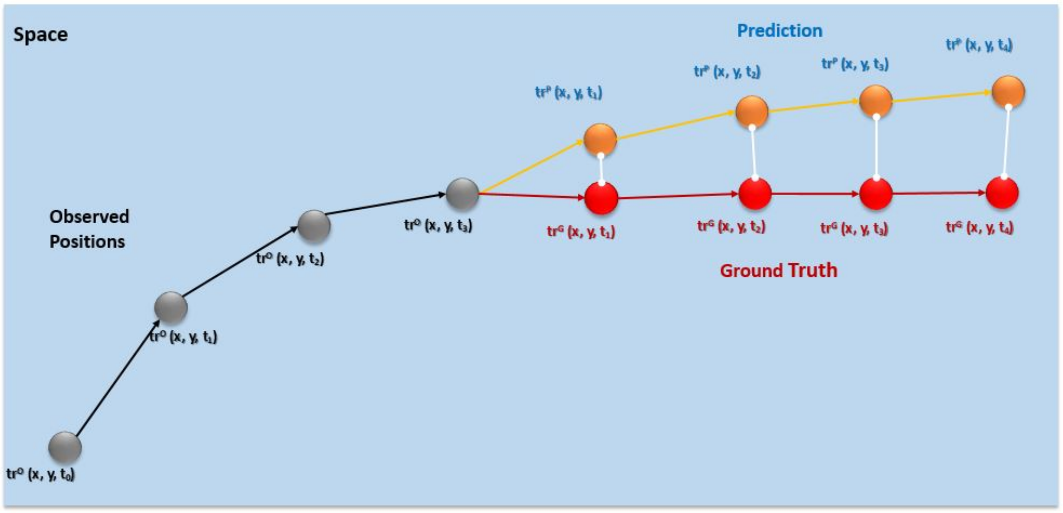


Figure 2.24: Average Displacement Error.

- **Final Displacement Error (FDE):** is the difference between the predicted final destination and the true final destination,as illustrated in the Figure 2.25.

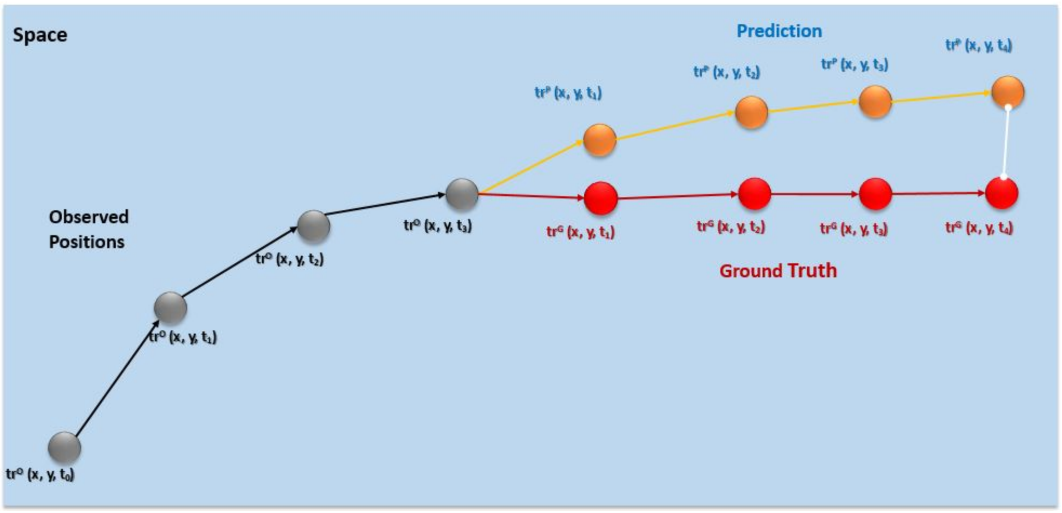


Figure 2.25: Final Displacement Error.

While the definition of both metrics is clear, the calculation of them have varied over time on the research works made on path prediction. In [57] as ADE they use the Mean Square Error (MSE) over all estimated points. Similar, in [97], for ADE they use the MSE, while for FDE they use the Euclidean distance. In [106], as ADE they use the Root of the Mean Squared Error (RMSE), and as FDE they mention the use of a distance but they do not specify which one, as in [57]. For this reason, the metrics used in the first part of this research work are shown in terms of the MSE for both ADE and FDE, as shown in the equation 2.11 and equation 2.12. Lately, in some works as in [109] and some challenges as the one for trajectory prediction given by the apolloScapes, the Euclidean distance is used for both FDE and ADE. For that reason the results shown in the final chapters are given using the Euclidean distance, as shown in the equation 2.13 and equation 2.14.

$$ADE = \frac{\sum_{i=1}^{n}\sum_{t=1}^{tpred}\left[(\hat{x}_i^t - x_i^t)^2 + (\hat{y}_i^t - y_i^t)^2\right]}{n(tpred)} \tag{2.11}$$

$$FDE = \frac{\sum_{i=1}^{n}\left[(\hat{x}_i^{tpred} - x_i^{tpred})^2 + (\hat{y}_i^{tpred} - y_i^{tpred})^2\right]}{n} \tag{2.12}$$

$$ADE = \frac{\sum_{i=1}^{n}\sum_{t=1}^{tpred} \sqrt[2]{(\hat{x}_i^t - x_i^t)^2 + (\hat{y}_i^t - y_i^t)^2}}{n(tpred)} \tag{2.13}$$

$$FDE = \frac{\sum_{i=1}^{n} \sqrt[2]{(\hat{x}_i^{tpred} - x_i^{tpred})^2 + (\hat{y}_i^{tpred} - y_i^{tpred})^2}}{n} \tag{2.14}$$

where $(\hat{x}_i^t, \hat{y}_i^t)$ are the predicted positions of the tracklet i at time t, (x_i^t, y_i^t) are the actual position (ground truth) of the tracklet i at time t, $tpred$ is the final destination or the last prediction step, and n is the number of tracklets in the testing set.

ADE is required when we need to know the position of an object at each time step, this information is important in application such as path planning, assistive robotics. FDE is important when we only need to know the final position of the object, this information is important in application such as collision avoidance systems. However, in the field of autonomous vehicle both FDE and ADE can be used depend on the task to be executed.

Finally, we adopted the use the weighted sum of ADE (WSADE) and weighted sum of FDE (WSFDE) as metrics as shown in [1], which are presented in the equation 2.15 and equation 2.16.

$$WSADE = D_v ADE_v + D_p ADE_p + D_b ADE_b \tag{2.15}$$

[1] http://apolloscape.auto/trajectory.html

$$WSFDE = D_v FDE_v + D_p FDE_p + D_b FDE_b \tag{2.16}$$

2.6.1 Limitations

Even though, FDE and ADE metrics are used in this research work, in seem right to point out some of their limitations. The limitations listed below are mainly for ADE, since FDE only take into account the last position of a path in the measurement as shown in Figure 2.24 and Figure 2.25:

- Very strict evaluation of the distance between the set of tracks that form a paths. As illustrated in Figure 2.24, ADE measures the distance between two paths tr^p and tr^G track by track. However, what happen if only one or two of the tracks in tr^p is really far from their corresponding track in tr^G, this means that the whole tr^p will be evaluated with a high error in the prediction.
- As mentioned in [86], which presents an study about trajectory analysis, another limitation of ADE, which is based in the Euclidean Distance, is that it is necessary to have paths of the same length in order to be performed, Table 2.1. In a real path prediction problem, the paths generated by objects are not of the same length, which pose a challenge for comparing them.

Overcoming those limitations will allow us to create flexible models for path prediction. So it would be interesting to evaluate the metrics shown in table 2.1 as future work.

2.7 Discussion and Conclusions

Path prediction task is important in different application and has been addressed using different approaches. In the literature, it was observed that path prediction can be addressed as a time series forecasting problem so considering that fact two main approaches were identified to process sequential data as time series, The Kalman

Table 2.1: Summary of common distance measurements.

Measurement	**Unifying lengths**	**Computational complexity**
Euclidean	Yes	O(n)
Hausdorff	No	O(mn)
Bhattacharyya	Yes	O(n)
Frechet	No	O(mn)
Longest Common Subsequence (LCSS)	No	O(mn)
Dynamic Time Warping (DTW)	No	O(mn)

Filter (KF) and the LSTMs (Long Short-Term Memory Architectures). Also, a trend in using more information than only (x, y) position of the object was identified on the literature review. This trend motivates in this research work to the development of a holistic path prediction approach able to process, besides x, y positional information of the objects, more information of the traffic scenes where the objects are moving. These extra set of information or contextual information such as visual information of the object, ego-vehicle features, other objects positions and visual information of the whole scene can be leveraged to improve performance in the path prediction against using only x, y positional information of the objects.

Most of the works use datasets where labeled data was not available so they have to tackle the problem of Object Detection and Tracking in addition to path prediction. KITTI was the only dataset that included labeled data of Detected and Tracked objects along with more data of the ego vehicle and the scene. All this from the perspective of cameras mounted on a vehicle, which is the main focus of this research, so because of that mainly KITTI was adopted in the work reported in this research thesis. In the literature of path prediction it was also observed the use of two metrics Average Displacement Error (ADE) and Final Displacement Error (FDE) so those two metrics were adopted to report the results.

Chapter 3

LSTMs for Single Path Prediction: Baseline Methods and Proposed Experimental Methodology

3.1 Introduction

As mentioned in the previous chapter, a path P is a set of tracklets, tr, that contains information such as $tr(x, y)$ observed position (coordinates) of an object that travels a given space, $P = \{tr_{t1}, tr_{t2}, ..., tr_{tlength}\}$. Each tr is a measure given for a sensor in intervals of time and in an ordered manner, $tr(x, y, time)$. This means that a path is a sequence of measurements of the same variable collected over time, where the order matters, resulting in a time series. This means that by analysing the measurements, in this case the observed positions of an object, sequentially it is possible to predict the future position of an object. Because of this, a path can be seen as a multivariate time series that has two time-dependent variables x, y. Each variable depends on its past values, and this dependency is used for forecasting future values. So the task of path prediction can be seen as forecasting a multivariate multi-step time series.

Regarding the literature review in the previous chapter, two main approaches were identified to process sequential data, The Kalman Filter (KF) and LSTMs (Long Short-Term Memory architectures). The KF updates its internal parameters and state of the system according to the input data sequence and learns from each data in the sequence to give an output relate to the given sequence. LSTM architectures are also able to process sequences of data, its internal memory cell can learn long and short term dependencies of a sequence of data and taking into account those dependencies gives an output related to the input sequence. Because of the capacity to process sequential data, the main focus of this chapter is to evaluate the performance of the KF and LSTMs to predict the future position of objects that are normally present in traffic scenarios, such as pedestrians, vehicles and cyclists, for different time prediction horizons using the KITTI dataset.

KITTI is selected because of its realistic scenes, such as highways, inner city, vehicles standing, vehicle moving, its different objects and the labeled data in image coordinate and 3D information. KITTI provides all this information from the perspective of cameras mouthed on a vehicle which is the main focus of this research work. We apply the prediction from two perspectives: image coordinate (pixels) and birds-eye view (metres). Image coordinate is the most common data used in published datasets while a birds-eye view, which measures in real-world distance values, is a more realistic measurement.

The experiments performed in this chapter are related to RQ1, since they show the exploration and results of representing the position of objects as time series and also as Relative Tracklet Position (RTP).

This chapter contains partial information from work published (peer-reviewed) in the IPTA 2019 conference. The remainder of this chapter is structured as follows: Section 3.2 Data Definition, explains the nature of the data used; Section 3.3 presents our approach; Section 3.4 and 3.5 present the experimental setup and results respectively; in section 3.6 a conclusion is given.

3.2 Data Definition

In this work we apply a sliding window over one track per time period then these smaller segments are split into two vectors of equal size. The first vector is the observed tracklets $tr^O = [tr^O_{t1}, tr^O_{t2}, ..., tr^O_{tobs}]$ and the second vector is its respective ground truth tracklet $tr^G = [tr^G_{t1}, tr^G_{t2}, ..., tr^G_{tpred}]$. The predicted vector of each tr^O is called $tr^P = [tr^P_{t1}, tr^P_{t2}, ..., tr^P_{tpred}]$. The aim of this chapter is to predict a tr^P based on the observed tracks tr^O, as illustrated in Figure 3.1.

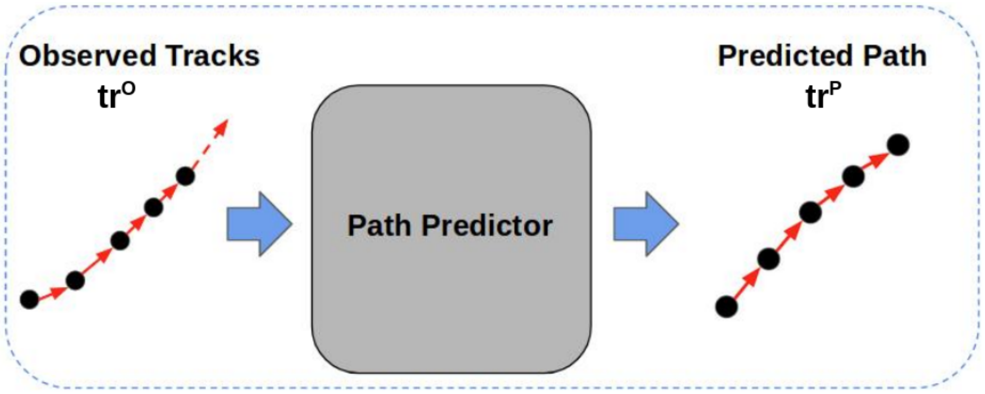


Figure 3.1: Predicting one path.

3.3 Approach

This section presents the specific methodology followed in this research thesis to develop a single-shot approach which leverages an LSTM architecture to predict the future path of moving objects normally present in traffic scenes such as pedestrians, vehicles and cyclists.

LSTMs have shown good performance when dealing with time series and so in this approach an LSTM architecture is used for path prediction. LSTMs can be used in different manners, two of these are Recursive Multi-step Forecast and Multiple Output Strategy.

Recursive Multi-step Forecast uses a one-step model time by time, where the prediction from the prior time step is used as an input for making a prediction on the following time step, as illustrated in Figure 3.2. Specifically for path prediction, this can be seen as the generation of a path step by step. This approach can be used as follows:

1. $Input = [tr_{t1}, tr_{t2}, ..., tr_{tobs}]$
2. $ptr_{t1} = model.predict(Input)$
3. $Input = [tr_{t2}, ..., tr_{tobs}, ptr_{t1}]$
4. $ptr_{t2} = model.predict(Input)$
5. $Input = [..., tr_{tobs}, ptr_{t1}, ptr_{t2}]$
6. $ptr_{t3} = model.predict(Input)$

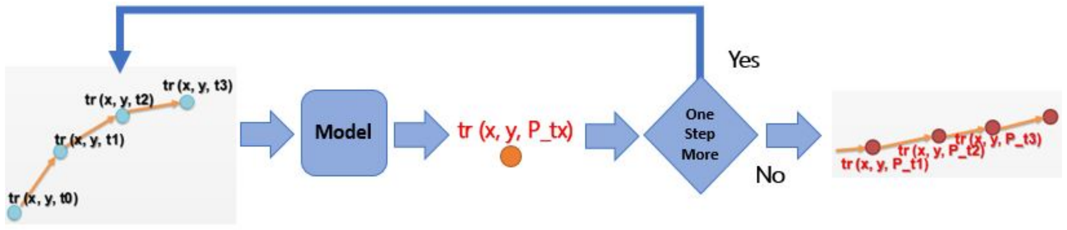


Figure 3.2: Recursive Multi-step Forecast.

This process is repeated *tpred* times, where *tpred* is the number of tracks or steps to predict ahead.

Multiple Output Strategy develops one model to predict an entire sequence in a one-shot manner, as presented in Figure 3.3. Like other types of neural network models, the LSTM can output a vector directly that can be interpreted as a multi-step forecast. This approach can be used in the following way:

1. $Input = [tr_{t1}, tr_{t2}, ..., tr_{tobs}]$
2. $Output = model.predict(Input)$
3. $Output = [ptr_{t1}, ptr_{t2}, .., ptr_{tpred}]$

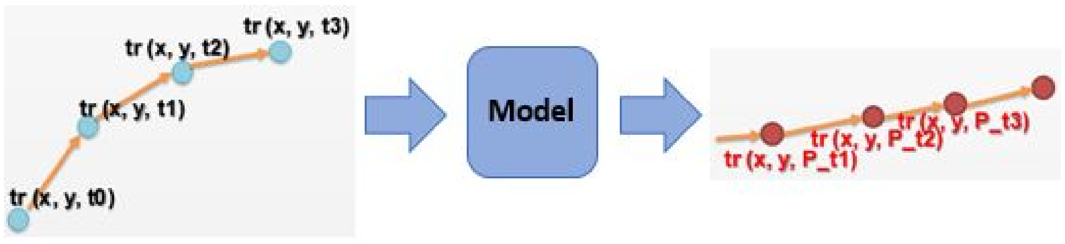


Figure 3.3: Multiple Output Strategy.

In this work, the multiple output strategy was adopted. The reason of selecting this approach is that at this time in the whole research our objective was to predicting a whole trajectory at once which makes the prediction phase faster than using the Recursive Multi-step Forecast approach. To use an LSTM in this manner the input and ground truth (GT) output data were configured as:

$$\begin{aligned} &Input\ data &&= [NSamples, tobs, Features] \\ &GT\ output\ data &&= [NSamples, PSize] \end{aligned}$$

where $NSamples$ is the number of samples that constitute the training data. $tobs$ is the size of tracklets used for predicting, i.e., 5 tracks to predict 5 steps ahead. $Features$ is the number of variables that constitute each track. In this case two features were used, the position (x, y), and $PSize$ is the number of outputs in the prediction. As the last dense layer can only be a one dimensional array, this can be calculated as $tpred * Features$ in the GT output data.

3.3.1 Model Architecture

Due to more availability of documentation, the Keras API [1] was used for the implementation of the LSTM architecture. To select the parameters a grid search was executed over the whole dataset, including all objects, and the following configuration achieved the best result. One layer was selected since adding more layers does not improve performance, as shown in [66], where they also mention that due to their recurrent nature, even a single layer of LSTM nodes can be

[1]https://keras.io/

considered as a “deep” neural network:

- **Number of layers:** 1.
- **Number of neurons:** 128.
- **Loss:** MSE
- **Optimizer**: Adam.

3.4 Experimental Setup

A first question to answer in this research work is: **Q1** ***How should the observed object position (tracklets) be best represented?***. Regarding this question, this section details the steps followed to evaluate the two selected approaches on the KITTI dataset by representing the positional information of the objects as time series and also as Relative Tracklet Position (RTP). RQ1 is initially explored in this chapter, a further study is performed in chapter 6.

As mentioned in chapter 2, section **Traffic Datasets**, several datasets can be found in the literature that can be used for path prediction. However, each datasets has its limitation, [60] do not provide annotated tracklets of the object, [13, 19] only provide data of one type of object, [13, 14, 19] provide data in only one perspective which is not from a camera mounted on a vehicle. Because of that KITTI [26] was selected since is the dataset that provides the positional information (labeled data in image coordinate and 3D information) and tracking labels of three types of objects – pedestrian, vehicles and cyclists, along with other features of the scene which will be leveraged further in this work. KITTI also has realistic traffic scenes, such as highways, inner city, vehicles standing, vehicle moving. More importantly, KITTI provides all this information from the perspective of cameras mouthed on a vehicle which is the main focus of this research work. Furthermore, at the time of developing these experiments KITTI was a widely used dataset in the literature of autonomous vehicle research such as object detection, tracking and depth estimation.

First, the methodology followed to extract the path of the object from the KITTI dataset is explained. Next, the two evaluated approaches are described. Then, the steps followed to train the LSTM are given. Finally, the process applied to the output of the model to get the predicted paths is explained.

3.4.1 Data Pre-processing

A crucial phase of dealing with time series prediction is understanding and preparing the data. In this chapter, the KITTI dataset was used and pre-processed as follows:

1. The KITTI data set was parsed to a simpler format with each track described by seven features: [Frame Number, Object Type, ID, XMin, YMin, XMax, YMax].
2. Trajectories were extracted for each object per sequence. KITTI contains several type of objects such as pedestrians and vehicles. This step is necessary to not mix tracks of other objects.
3. Tracklets (sub-trajectories) were created of a certain consistent length. For each object trajectory, tracklets of size 10, 20, 30 and 40 tracks were extracted, these tracklets constitute the data set used for training and testing. This was done because the approaches are tested to predict 5, 10, 15 and 20 steps/frames in the future. The half of each tracklet is used as observed tracklet and the other half as ground truth.
4. The center of the bounding boxes for the objects were extracted.
5. The tracklets were translated to relative positions. This process consists of setting the first (x, y) position of each tracklet to $(0, 0)$ and all the following tracks are adjusted relative to this point. Tracklets that have been adjusted will be referred to as Relative Tracklet Position (RTP) and the original, unadjusted tracklets, as Absolute Tracklet Position (ATP). Experiments were done for both type of tracklets.

A total of 853 objects were extracted from all the sequences – 167 pedestrians, 649 vehicles and 37 cyclists – for the four prediction horizons of 5, 10, 15 and 20 frames. The number of tracklets are shown in Table 3.1 for each object on the four prediction horizons (P.H.).

Table 3.1: Size of the training data for all four prediction horizons.

Number of Tracklets				
P.H.	Pedestrian	Vehicle	Cyclist	All
±5	6,933	18,662	1,282	28,138
±10	5,933	14,591	1,028	22,907
±15	5,076	11,365	831	18,714
±20	4,371	9,172	665	15,658

3.4.2 Comparative Study

From the literature review, two approaches were identified to process sequential data. The Kalman filter (KF) and Long Short-Term Memory architectures (LSTMs). The Kalman filter is an algorithm that uses a series of data observed over time to estimates future measurements with more accuracy [50]. Because of its efficiency the Kalman filter has been used in several application such as navigation [11, 59], object tracking [22, 32] and path prediction [33, 40]. Because of its wide application and availability of information to understand its algorithm, the Kalman filter is used as a baseline in this research. However due to its limitation to process different type of information its exploration led us to look for more complex techniques, specifically Long Short-Term Memory architectures (LSTMs).

LSTM architectures are currently used in areas such as language translation [45, 51], time series prediction [64, 76] and trajectory prediction [85, 57, 66, 84, 74]. LSTMs are capable of getting long and short term dependencies from sequences and then based on those dependencies predicting future information. To predict the future path of an object in a scene, we have to analyse the previous information of such object, i.e. previous x, y positions (tracklets). Due to the characteristics of path prediction data, LSTM architectures are suitable to tackle this problem, since this type of networks are able to process sequential data and leverage information through time, and based on that perform a better prediction.

The application of LSTM architectures on sequential data such as time series forecasting and language translation problems was novel at the beginning of this research and its performance on path prediction still required further exploration. Because of that, to understand the potential of LSTMs for path prediction in the context of moving cameras in traffic scenes and to establish the way of using them in a model, an LSTM-based model is evaluated against the Kalman Filter. We compare our approach with one baseline methodology using the Average Displacement Error (ADE) 2.11 and Final Displacement Error (FDE) 2.12:

The Kalman Filter (KF): the KF was used with the Constant Velocity (CV) model. This model has shown good performance when dealing with linear movements.

Vanilla LSTM One-Shot(VLSTMOS): this consists of a one layer LSTM with 128 neurons. In the literature, when an LSTM is used with its basic configuration it is called Vanilla LSTM.

3.4.3 Training

During training, from the available data a random training and test split was done; 70% of the data was used for training and the rest for testing. Also, to feed the model correctly, the next steps were followed:

1. Specify the length of tracklets to be processed (10-40).
2. Scale data [0,1]: un-scaled input variables can result in a slow or unstable learning process, whereas un-scaled target variables on regression problems can result in exploding gradients causing the learning process to fail.
3. Split data into observed tracklet, tr^O, and ground truth (tracklet) path to be predicted, tr^G.

4. tr^O is shaped as $[Nsamples, tobs, features\text{-}in\text{-}tr^O]$ and tr^G as $[Nsamples, OutputLength]$. Where $Nsamples$ is the number of samples in the dataset, $tobs$ is the number of tracks in each observed tracklet (tr^O), $features\text{-}in\text{-}tr^O$ is the number of features in each observed track. Finally, $OutputLength$ is the size of the output of the model. In this case, $OutputLength = tpred * features\text{-}in\text{-}tr^G$. Here $tpred$ is the steps to predict in the future (the prediction horizon) and $features\text{-}in\text{-}tr^G$ is the number of features to predict in each step.

3.4.4 Single Path Prediction

For this phase, the output of the model was processed as follow:

1. The model outputs a single array tr^P per tr^O of size $OutputLength$.
2. tr^P is re-shaped to $[tpred, features - in - tr^G]$.

3.5 Results

This set of experiments evaluates the performance of a Kalman filter with Constant Velocity model (baseline) and an LSTM network on the KITTI dataset. For the LSTM results, two sets of experiments were executed. One where the tracklets keep their absolute position (ATP) and other where the tracklets are translated to a relative position (RTP). For the Kalman filter the same two set of experiments was carried out but the comparison is not shown since the results using ATP vs RTP showed no difference. The performance was calculated on four different prediction horizons (P.H.) and for three different objects – pedestrians, cyclists and vehicles (labeled as Cars, Vans and Trucks). The data for training and testing consists of the center of the object. The results are also provided in image coordinate (pixels) and in birds-eye view (metres). Due to the size of the image (1224 × 370 pixels), the results show large values in the case of image coordinates. Fig. 3.4 illustrates the approximate real world implication of variations in pixels as applied to the KITTI dataset. Finally, the relative improvement (error reduction) values shown in

Table 3.4 and 3.5 were calculated by $[(newValue - originalValue)/originalValue] * 100$ where $newValue$ is the LSTM RTP methodology and $originalValue$ is the approach to be compared with. The negative values means that there was an error reduction using the LSTM RTP approach.

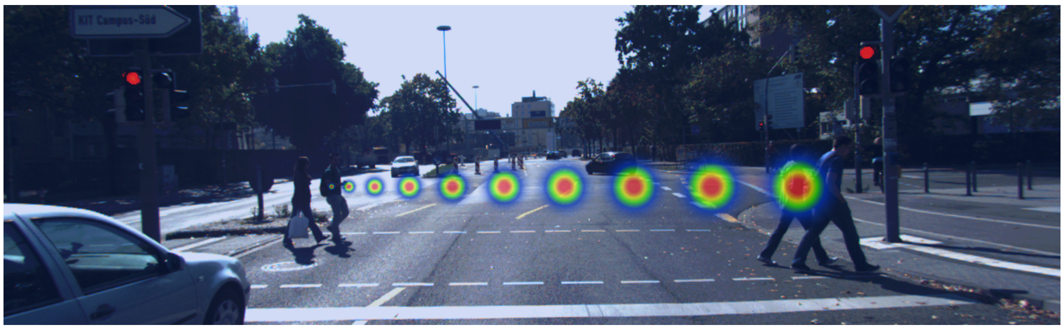


Figure 3.4: Heat maps of 10-100 pixels (left to right) illustrating pixel differences in the real world.

3.5.1 The Kalman Filter vs LSTM RTP

Table 3.2, in columns three and four, shows the results on image coordinates obtained by the Kalman Filter and LSTM RTP and Table 3.4 depicts the improvement comparing both approaches.

Table 3.3, in the columns three and four, shows the results on birds-eye view obtained when executing the Kalman Filter and LSTM RTP, whilst table 3.5 presents the improvement of LSTM RTP over the Kalman Filter. As in table 3.4, the negative values indicate that there was error reduction using LSTM RTP.

The results show that LSTM RTP outperforms the simple Kalman Filter on most cases, specifically for vehicle and pedestrians. Also, it can bee seen that LSTM RTP performs better when predicting in birds-eye view compared to using image coordinates.

3.5.2 LSTM ATP vs LSTM RTP

Table 3.2, in the columns two and three, shows the results on image coordinates obtained when executing the LSTM using the two different type of tracklets (ATP

Table 3.2: Path Prediction accuracy using Image Coordinates

Image Coordinate (Pixels)									
Approach	**LSTM ATP**			**LSTM RTP**			**The KF**		
Metric	**ADE**			**ADE**			**ADE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	100	119	137	76	98	94	111	225	143
±10	375	297	3,242	244	353	160	259	585	287
±15	936	734	38,16	802	668	1163	549	1,019	807
±20	1,305	1,127	35,085	1,755	1,065	6,856	970	1,553	1,944
Metric	**FDE**			**FDE**			**FDE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	217	289	397	169	252	271	214	501	385
±10	988	947	7,189	727	1,163	538	778	1,914	1,022
±15	2,859	2,507	8,602	2,483	2,412	4,452	1,907	3,695	3,408
±20	4,219	4,114	15,7686	6,251	4,040	25,743	3,567	5,808	8,539

Table 3.3: Path prediction accuracy using birds-eye view (real world measures)

Birds-Eye View (metres)									
Approach	**LSTM ATP**			**LSTM RTP**			**The KF**		
Metric	**ADE**			**ADE**			**ADE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	0.026	0.085	0.124	0.007	0.065	0.023	0.062	0.465	0.236
±10	0.048	0.332	0.484	0.051	0.248	0.105	0.104	0.752	0.356
±15	0.120	1.184	1.021	0.121	0.872	0.413	0.213	1.122	0.544
±20	0.240	2.027	4.580	0.219	1.679	1.011	0.412	1.791	0.904
	FDE			**FDE**			**FDE**		
P. H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	0.052	0.179	0.254	0.016	0.153	0.043	0.075	0.575	0.272
±10	0.131	0.953	1.271	0.145	0.721	0.271	0.246	1.666	0.637
±15	0.385	3.290	2.745	0.378	2.685	1.289	0.688	3.285	1.416
±20	0.732	6.660	11.219	0.767	5.479	3.062	1.490	6.036	2.953

and RTP) and Table 3.4 depicts the relative improvements.

Table 3.3, in the columns two and three, shows the results on the birds-eye view obtained when executing the LSTM using the two different type of tracklets (ATP and RTP), whilst Table 3.5 presents the improvement.

This set of results show clearly that there was error reduction in most of cases when translating the tracklets to relative position (RTP). In a few cases, mostly pedestrians, a slight increase in error was found.

Table 3.4: Improvements using image coordinates

Image Coordinate (Pixels) Improvement %						
	LSTM ATP vs LSTM RTP			**The Kalman Filter vs LSTM RTP**		
Metric	**ADE**			**ADE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	-24.22	-17.24	-31.32	-31.77	-56.25	-34.43
±10	-34.92	19.05	-95.06	-5.94	-39.58	-44.27
±15	-14.23	-9.05	-69.51	46.14	-34.49	44.11
±20	34.51	-5.46	-80.46	81.00	-31.41	252.76
Metric	**FDE**			**FDE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	-22.37	-12.63	-31.69	-21.06	-49.60	-29.64
±10	-26.37	22.73	-92.52	-6.46	-39.26	-47.37
±15	-13.13	-3.79	-48.24	30.19	-34.73	30.65
±20	48.15	-1.81	-83.67	75.26	-30.44	201.47

To have a better understanding of the quantitative results, a visualization of the predicted paths was created. Visualizations of results in image coordinates are shown in the figures 3.5 , 3.6 , 3.7 , 3.8 for the four time prediction horizon and for the object vehicle. The first 1,000 samples were printed. The first image (GT) displays the ground truth paths and the rest present the paths predicted by each model. If the ADE and FDE of each approach were close to zero their predicted paths shown in the RTP, ATP and KF image should be similar or equal to the GT image. However as the ADE and FDE of each approach increase the dissimilarity of their paths also increase compared to the GT Image. Regarding that, for each

Table 3.5: Improvements in Bird-Eye View.

Bird-Eye View (Meters) Improvement %						
	LSTM ATP vs LSTM RTP			**The Kalman Filter vs LSTM RTP**		
Metric	**ADE**			**ADE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	-74.04	-24.05	-81.80	-89.19	-86.05	-90.40
±10	8.04	-25.19	-78.35	-50.76	-66.97	-70.52
±15	0.52	-26.35	-59.61	-43.07	-22.26	-24.10
±20	-8.74	-17.16	-77.92	-46.77	-6.26	11.78
Metric	**FDE**			**FDE**		
P.H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	-68.13	-14.30	-83.02	-78.14	-73.39	-84.16
±10	10.25	-24.35	-78.64	-41.30	-56.73	-57.41
±15	-1.86	-18.38	-53.05	-45.06	-18.24	-8.97
±20	4.82	-17.73	-72.71	-48.52	-9.22	3.71

approach, it can be seen that for large PH the error increase and the predicted paths diverge from the ground truth. The same happens for the results in BEV, which are visualized in the figures 3.9, 3.10, 3.11, 3.12. In both cases, image coordinates and BEV, the paths predicted by LSTM using RTP are more similar to the paths in GT in most of the PH.

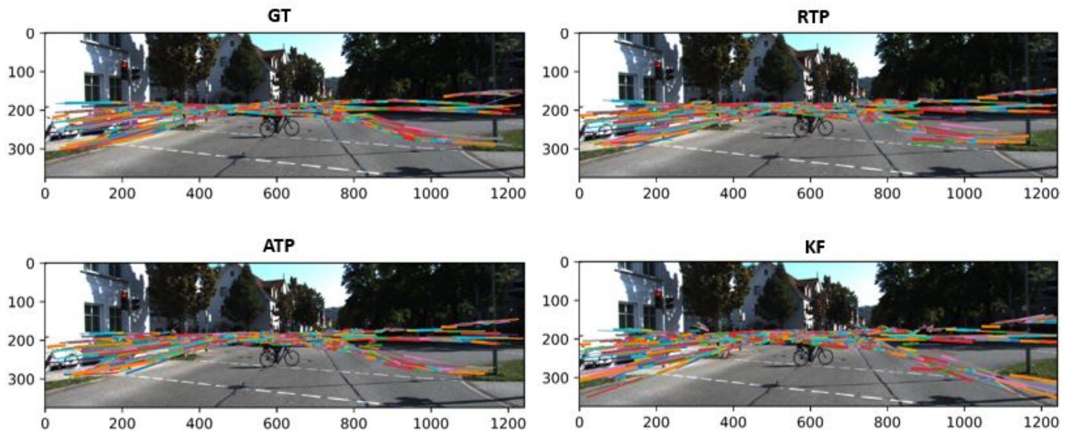


Figure 3.5: IPTA Img Vehicle 0.5Sec

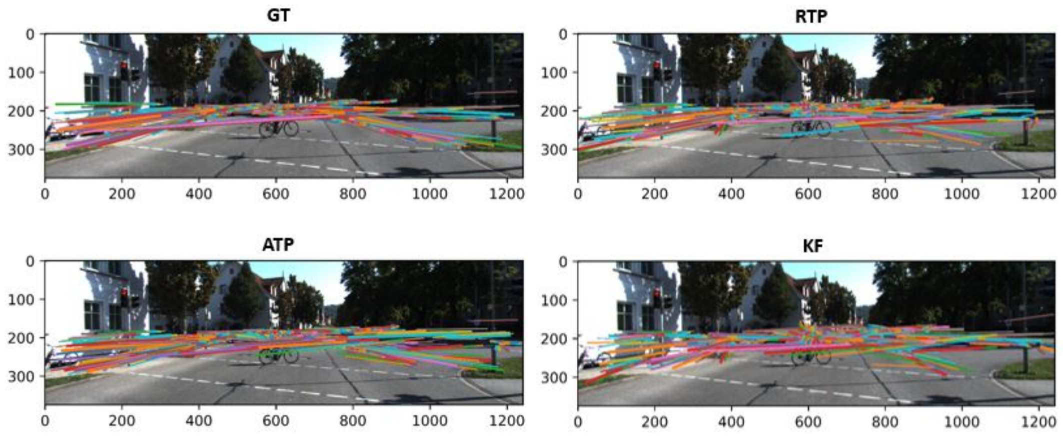


Figure 3.6: IPTA Img Vehicle 1 Sec

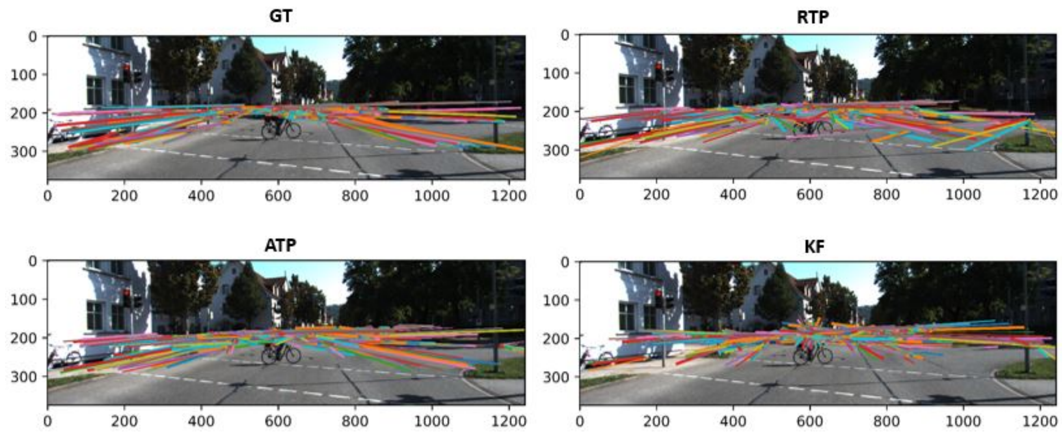


Figure 3.7: IPTA Img Vehicle 1.5 Sec

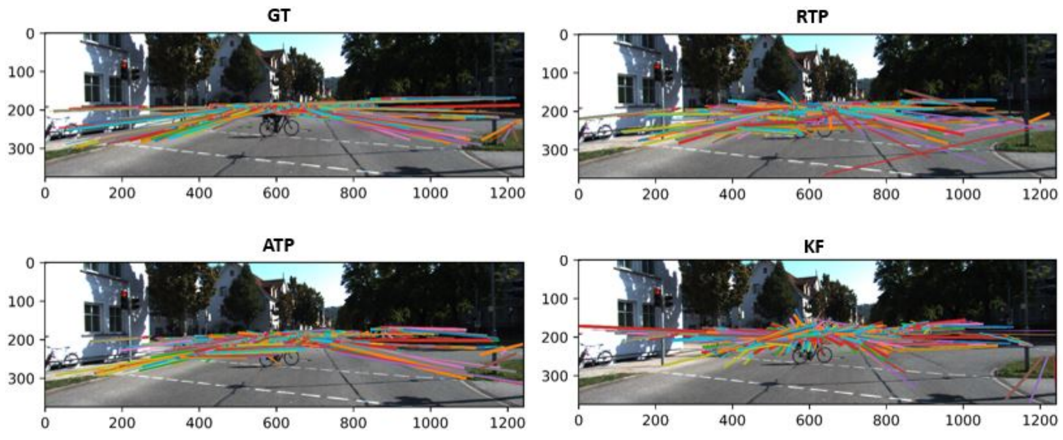


Figure 3.8: IPTA Img Vehicle 2 Sec

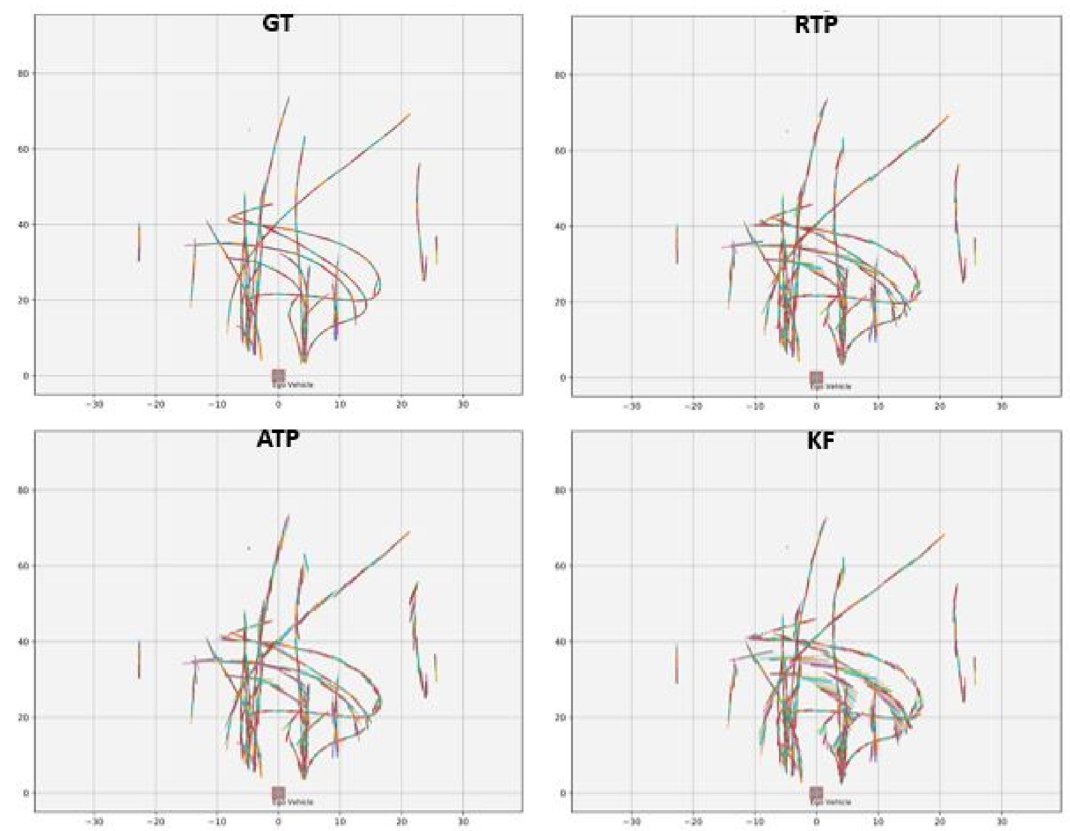


Figure 3.9: IPTA BEV Vehicle 0.5Sec

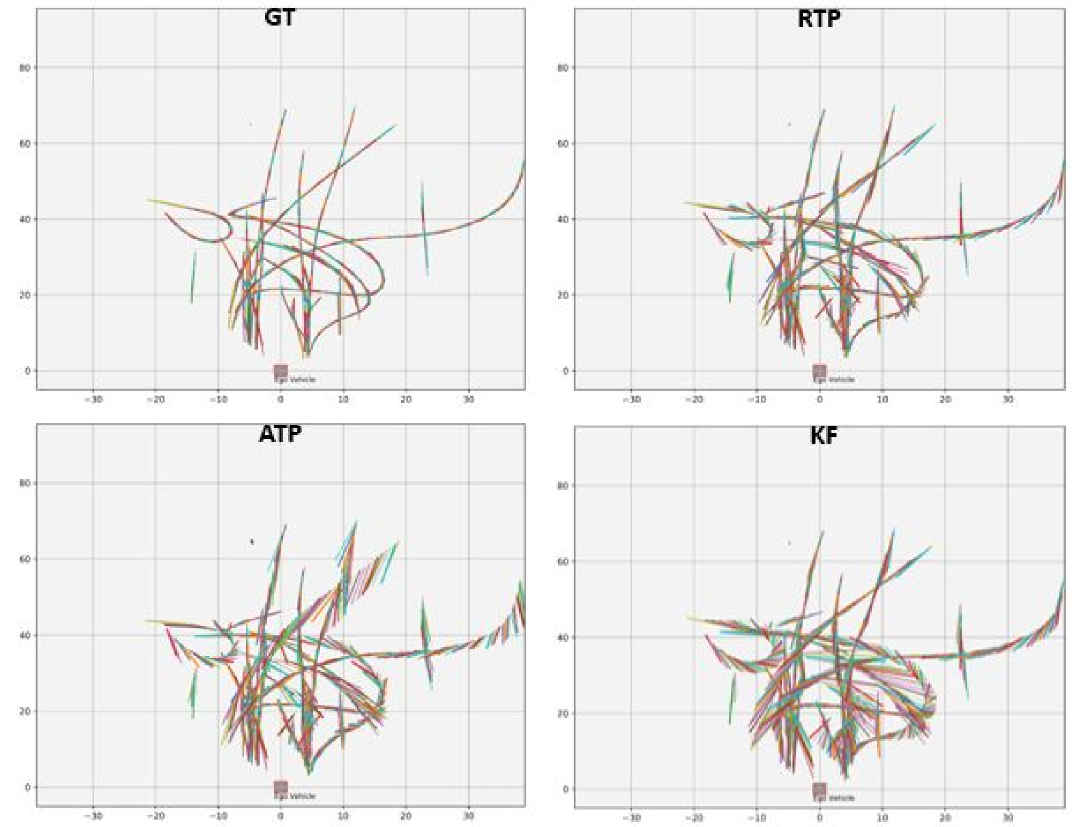


Figure 3.10: IPTA BEV Vehicle 1 Sec

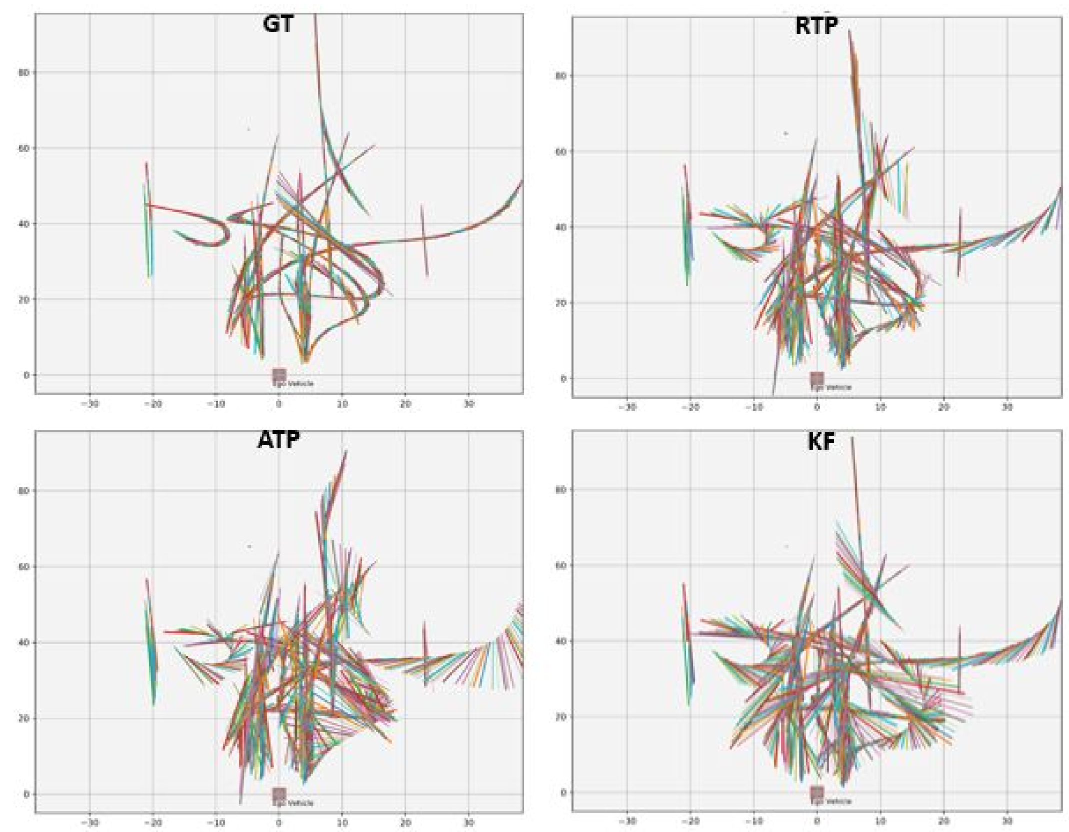


Figure 3.11: IPTA BEV Vehicle 1.5 Sec

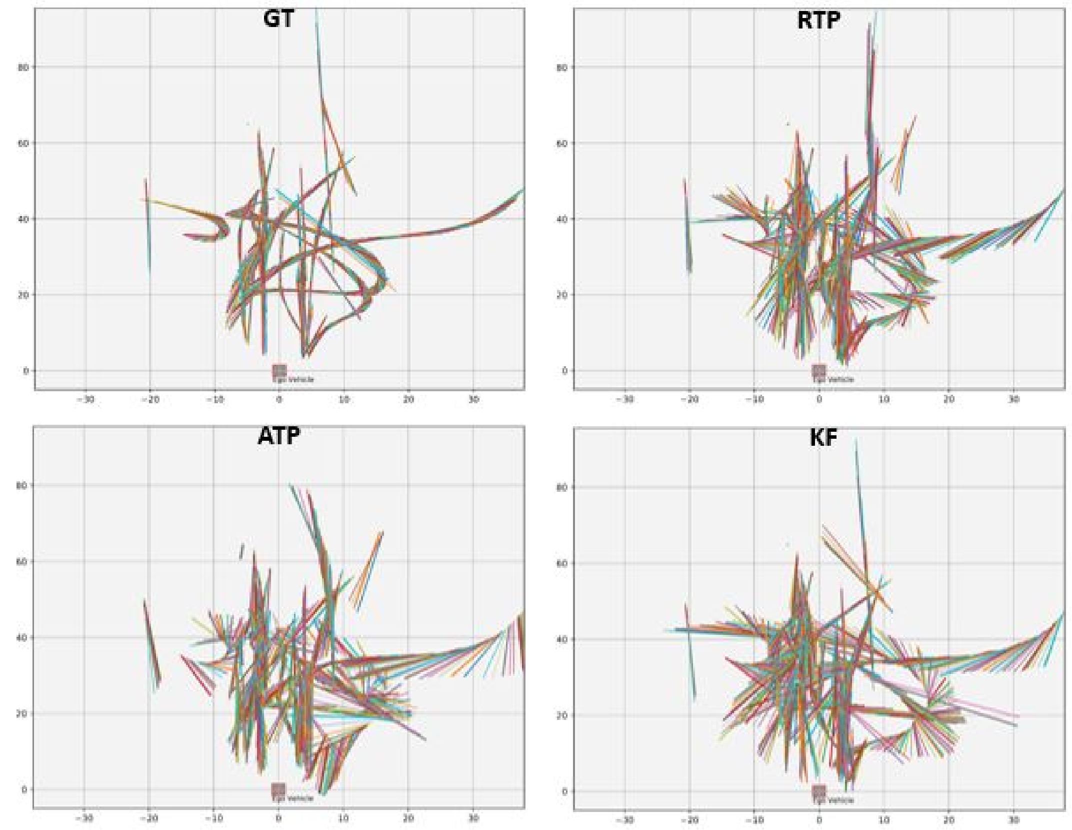


Figure 3.12: IPTA BEV Vehicle 2 Sec

3.6 Discussion and Conclusions

This chapter presents a single-shot prediction approach that uses one LSTM to predict the future position of objects commonly present in traffic scenes. The selected data set was KITTI because of its realistic scenes, such as highways, inner city, vehicles standing, vehicle moving, its different objects and the labeled data in image coordinate and 3D information. The objective of this work was to compare the performance of the commonly used Kalman Filter (baseline) with the newer options offered by LSTM architectures and analyze some of the potential influences on their trajectory prediction accuracy by looking at three object classes, four prediction horizons and two different perspectives (image coordinate and birds-eye view).

The results obtained in this chapter show that LSTM approaches perform well for predicting the near future paths of objects in the context of cameras mounted on a moving vehicles. It shows also that the performance of this approach is affected by the prediction time horizon, the largest the prediction horizon resulting in the largest errors.

For the LSTM architecture, it can be clearly seen that translating the tracklets to a relative position (RTP) helps the model to learn. The reason for this could be that, RTP makes the tracklets to be similar in that space and produces more examples for learning. Another important point to note is that predicting in birds-eye view is better than predicting using image coordinate, however 3D information is not always available.

The results also indicate that this approach is affected by the size of the training data. For instance, for the class Vehicle, LSTMs outperforms the Kalman Filter for all prediction horizons. One probable reason is that there is a bigger dataset to train for this object class , while for the other objects the size of the training data is small.

The results have shown that using an LSTM achieves good performance for a

P.H ±5 of up to an ADE of 0.01m for pedestrians, 0.06m for vehicles and 0.02m for cyclists and up to a FDE of 0.016m, 0.15m, 0.04m for the same objects improving the performance of the baseline Kalman Filter. The prediction horizon where the approach is most reliable is for ±5 and ±10 for image coordinate and for ±5 to ±15 for birds-eye view perspective.

Finally, the processing inference time per tracklet for the LSTMs approach was 0.02 ms/tr, 0.032 ms/tr, 0.045 ms/tr, 0.057 ms/tr for prediction time horizon (P.H.) of ±5 to ±20 respectively. The Kalman Filter processing time was 3.627 ms/tr, 6.961 ms/tr, 11.012 ms/tr, 13.553 ms/tr for PH of ±5 to ±20 respectively. All this using a computer with the following features: GPU GeForce GTX 980, CPU Intel® Core^{TM} i5-4690K CPU @ 3.50GHz x 4, RAM 24GB.

Chapter 4

Single Path Prediction: LSTMs Variants

4.1 Introduction

To understand the potential improvement in path prediction that a holistic approach can bring, we first look at the performance of a range of approaches to establish a baseline performance without using more features. In addition, the literature about LSTMs networks describes a number of alternative architectures that seek to outperform Vanilla LSTM [49, 61]. Therefore this chapter explores the variants of the Vanilla LSTM called stacked, bidirectional, return sequence, Gated Recurrent Unit (GRU), LSTMs as encoder-decoder and adding attention layers. A Conv1D model was also evaluated as an alternative method for processing time series data. These architectures were selected because they process the input data in different manners that a vanilla LSTM. This study can be seen as a second stage in the exploration of LSTMs because it allows us to understand better how to use LSTM architectures in its different ways, since this is needed in the next chapters where more complex model are designed for predicting multiple paths and for processing multimodal data. The purpose of this chapter is to present the performance of these models for the path prediction task using only positional information (x, y) of objects in birds-eye view (BEV) from the KITTI dataset. As mentioned in the previous chapter, KITTI is selected because of its realistic scenes, such as highways, inner city, vehicles standing, vehicle moving, its different objects and the labeled data in image coordinate and 3D information. All this information from the perspective of cameras mouthed on a vehicle which is the main focus of this research. However, only BEV was used because in this perspective the measurements are given in metres which represents better the position of the object in the real life than pixels in image coordinates.

This chapter is related to RQ2, by exploring several variants of LSTMs and the way the input data is processed by each of them, it helped to understand better the LSTM architectures and the knowledge obtained contributed to the creation of chapter 5 and chapter 6 where more complex models were used.

In the remainder of this chapter, Section 4.2 presents some related studies made on RNN architectures, Section 4.3 describes the models used in this study. Section 4.4 and 4.5 present the experimental setup and results respectively. Finally, conclusions are drawn in section 4.6.

4.2 Related Work

In [49] the authors aim to determine whether the LSTM architecture is optimal or whether better architectures exist. To do this, they conducted a thorough architecture search where they evaluated over ten thousand different RNN architectures, and identified architectures that outperform both the LSTM and the recently-introduced Gated Recurrent Unit (GRU) on three tasks, 1) Arithmetic, 2) XML modeling, and 3) Penn Tree-Bank (PTM). Finally, they added Music datasets to measure the generalization ability of the architecture search procedure (the music datasets was not used in the search procedure). In [61] they also perform an evaluation of the most popular LSTM architecture (vanilla LSTM) against eight different variants on three benchmark problems: acoustic modeling, handwriting recognition, and polyphonic music modeling. Each variant differs from the vanilla LSTM by a single change.

Both works [49, 61] conclude that none of the variants of LSTMs significantly outperform the LSTM. But most importantly, they determined that adding a positive bias to the forget gate greatly improves the performance of the LSTM. Given that this technique is the simplest to implement, they recommend it for every LSTM implementation. They found that adding a bias of 1 to the LSTM's forget gate closes the gap between the LSTM and the GRU. Adding a bias of size 1 significantly improved the performance of the LSTM on tasks where it fell behind the GRU and MUT1. Thus they recommend adding a bias of 1 to the forget gate of every LSTM in every application; it is easy to do and often results in better performance in those tasks. Interestingly, this idea has already been stated in the paper that introduced the forget gate to the LSTM [5]. This forget bias configuration is already included as a default in the Keras LSTM layer [1]

4.3 Models

The following architectures were selected because they process the input data in different manners that a vanilla LSTM. This study can be seen as a second stage in the exploration of LSTMs because it allows us to understand better how to use LSTM architectures in its different ways, since this is needed in the next chapters where more complex model are designed for predicting multiple paths and for processing multimodal data.

4.3.1 LSTM Vanilla

The Long Short-Term Memory (LSTM) unit was initially proposed by [3]. The LSTM Vanilla model consist of one LSTM layer with 128 units. The default values given by the keras LSTM layer is used. As shown in the table 4.1, the default configuration already includes the discovery made by [49, 61], where they recommend adding a bias of 1 to the forget gate at initialization. In the Keras API [2] this is done by setting unit_forget_bias=True.

[1]https://keras.io/api/layers/recurrent_layers/lstm/
[2]https://keras.io/

Table 4.1: LSTM Keras layer default configuration.

```
tf.keras.layers.LSTM(
units,
activation="tanh",
recurrent_activation="sigmoid",
use_bias=True,
kernel_initializer="glorot_uniform",
recurrent_initializer="orthogonal",
bias_initializer="zeros",
unit_forget_bias=True,
kernel_regularizer=None,
recurrent_regularizer=None,
bias_regularizer=None,
activity_regularizer=None,
kernel_constraint=None,
recurrent_constraint=None,
bias_constraint=None,
dropout=0.0,
recurrent_dropout=0.0,
implementation=2,
return_sequences=False,
return_state=False,
go_backwards=False,
stateful=False,
time_major=False,
unroll=False,
**kwargs
)
```

4.3.2 LSTM Vanilla Backwards

This model is similar to the Vanilla LSTM, the unique modification is that from the parameters shown in the table 4.1, the go_backwards parameter has to be set to $True$. This indicate to the model to process the input sequence backwards.

4.3.3 LSTM Bidirectional

Bidirectional RNN is another variant of RNN introduced by [4]. Similarly, bidirectional LSTM is an extension of an LSTM. Bi-LSTMs train two LSTMs layers instead of one on the input sequence, as illustrated in Figure 4.1. The first layer is trained on the input sequence in normal order and the second one on a reversed version. In other words, Bi-LSTMs runs the input sequence in two ways, one from past to future as normal LSTM and one from future to past. The results of both layers are then merged. This way it preserves information from past to future and from future to past. In this work the results were merged using the concatenated method. The Keras API was used to obtain the implementation of this model.

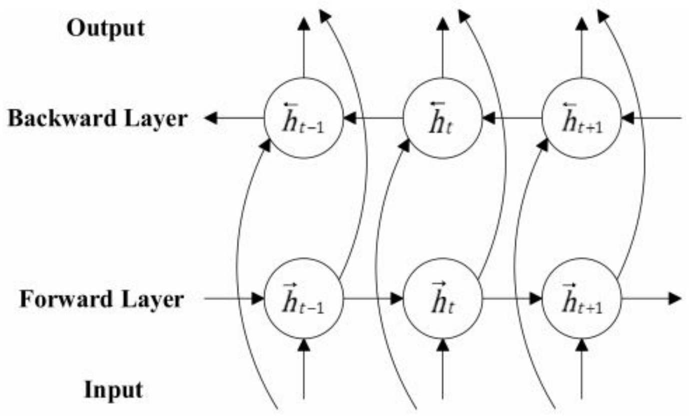

Output
Backward Layer
$\overleftarrow{h}_{t-1}$
$\overleftarrow{h}_{t}$
$\overleftarrow{h}_{t+1}$
Forward Layer
$\overrightarrow{h}_{t-1}$
$\overrightarrow{h}_{t}$
$\overrightarrow{h}_{t+1}$
Input

Figure 4.1: Bidirectional LSTM [83].

4.3.4 Stacked LSTMs

The Stacked LSTM was introduced by [28], where they found that stacking RNNs lead to better performance on speech recognition. RNNs are inherently deep in time, since their hidden state is a function of all previous hidden states. The question that inspired this paper was whether RNNs could also benefit from depth in space; that is from stacking multiple recurrent hidden layers on top of each other, just as feed forward layers are stacked in conventional deep networks. However in two more related works [66, 78], it is found that adding more layers to an LSTM model does not lead to significant improvement over a single LSTM layer. Actually, [78] clearly mentions the following, *"The increased model complexity combined with the variable performance suggests some degree of overfitting. By using fewer units and layers, the $R128by2$ model, which is the model using 2 stacked LSTMs with 128 units, maintained the advantages of learning long-term dependencies while limiting the model complexity and keeping it simple. This hypothesis is further supported by the outcomes described in the article, where $R128by2$ utilized fewer features than the more complicated models, allowing it to learn more general trends"*. Based on these studies, in this work we decided to use a model with two stacked LSTMs.

Stacked LSTMs consist of using more than one LSTM layer in a model. Different to using only one LSTM layer, the layer that connects with other LSTM layer has to return a sequence in order to be processed for the next one. Using the Keras API this can be done by only changing the parameter return_sequences=True, from the parameters shown in the table 4.1.

4.3.5 LSTM Encoder-Decoder

An RNN encoder-decoder was proposed by [38] for machine translation problems, this type of network consists of two RNNs. One RNN encodes a sequence of symbols into a fixed length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The same applies for LSTM encoder-decoder as used in [94] for predicting the trajectory of vehicles on a occupancy grid map.

The Encoder-Decoder LSTM can be implemented directly in the Keras deep learning library. One or more LSTM layers can be used to implement the encoder model. The output of this model is a fixed-size vector that represents the internal representation of the input sequence. The number of memory cells in this layer defines the length of this fixed-sized vector. The decoder must transform the learned internal representation of the input sequence into the correct output sequence.

One or more LSTM layers can also be used to implement the decoder model. This model reads from the fixed sized output from the encoder model. As with the Vanilla LSTM, a Dense layer is used as the output for the network. The same weights can be used to output each time step in the output sequence by wrapping the Dense layer in a Time Distributed wrapper.

There is a problem though, we must connect the encoder to the decoder, and they do not fit. That is, the encoder will produce a 2-dimensional matrix of outputs, where the length is defined by the number of memory cells in the layer. The decoder is an LSTM layer that expects a 3D input of [samples, time steps, features] in order to produce a decoded sequence of some different length defined by the problem. We can solve this using a Repeat Vector layer. This layer simply repeats the provided 2D input multiple times to create a 3D output. The Repeat Vector layer can be used like an adapter to fit the encoder and decoder parts of the network together. The Repeat Vector can configure to repeat the fixed length vector one time for each time step in the output sequence.

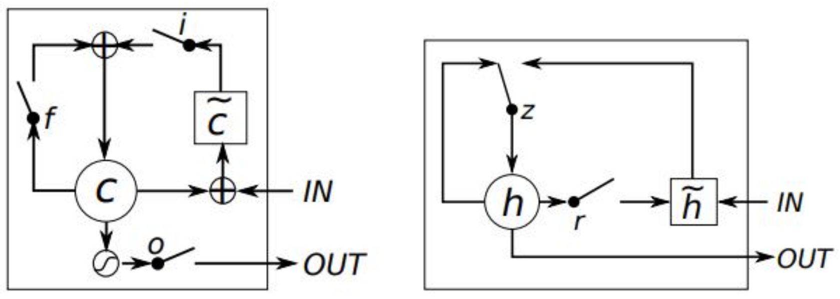
In this work the LSTM encoder-decoder consist of one LSTM layer as encoder and one LSTM layer as decoder.

4.3.6 GRU

The gated recurrent unit (GRU) was introduced by [38]. It can be said that is a simplified version of an LSTM network as show in Figure 4.2. [39] indicates some interesting difference between LSTM and GRU. One feature of the LSTM unit that is missing from the GRU is the controlled exposure of the memory content. In the LSTM unit, the amount of the memory content that is seen, or used by other units in the network is controlled by the output gate. On the other hand the GRU exposes its full content without any control. Another difference is in the location of the input gate, or the corresponding reset gate. The LSTM unit computes the new memory content without any separate control of the amount of information flowing from the previous time step. Rather, the LSTM unit controls the amount of the new memory content being added to the memory cell independently from the forget gate. On the other hand, the GRU controls the information flow from the previous activation when computing the new, candidate activation, but does not independently control the amount of the candidate activation being added (the control is tied via the update gate). In short, GRU is like an LSTM with a forget gate but has fewer parameters than LSTM, as it lacks an output gate.

In [39] they evaluate LSTM and GRU units on the tasks of polyphonic music modeling and speech signal modeling. Based on their experiments, they concluded that by using fixed number of parameters for all models on some datasets GRU, can outperform LSTM units both in terms of convergence in CPU time and in terms of parameter updates and generalization.

The work of [68] presents an evaluation of three variants of GRU units on the MNIST and IMDB datasets. Their final conclusion is that the three variant models perform as well as the original GRU RNN model while reducing the computational expense. Taking into account the conclusion of this work, in this experiments it was decided to use a simple GRU network.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 4.2: Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation [39].

4.3.7 LSTM Attention

Long input sequences can lead the model to a bottleneck situation in terms of modeling performance. This can be avoided by applying the attention mechanism proposed by [36]. The attention method extends the Seq2Seq model by informing the decoder of the positions of the elements in the encoder's input sequence where the most significant information could be located at each time step. This contributes to an improved performance [92]. The Encoder-Decoder architecture is popular because it has demonstrated state-of-the-art results across a range of domains. However, a limitation of this architecture is that it encodes the input sequence to a fixed length internal representation. This imposes limits on the length of input sequences that can be reasonably learned and results in worse performance for very long input sequences. Attention is the idea of freeing the encoder-decoder architecture from the fixed-length internal representation. This is achieved by keeping the intermediate outputs from the encoder LSTM from each step of the input sequence and training the model to learn to pay selective attention to these inputs and relate them to items in the output sequence. For that reason using attention layers in models has become a trend. It has been used in works such as machine translation [36], speech recognition [47], Image Captioning [54] and trajectory prediction [92]. In these experiments a model with one attention layer was used[3].

4.3.8 Conv1D

Motivated by the success of CNN architectures in various domains, researchers have started adopting them for time series analysis such as in [70]. Convolutional Neural Network (CNN) models were developed for image classification, in which the model accepts a three-dimensional input representing an image's pixels and color channels, in a process called feature learning. This same process can be applied to one-dimensional sequence of data. The model extracts features from sequences data and maps the internal features of the sequence. Here, a convolution can be seen as applying a filter over the time series (Convolution across time rather than space). Unlike images, the filters exhibit only one dimension (time) instead of two dimensions (width and height). The filter can also be seen as a generic non-linear transformation of a time series. For example, if convolution (multiplying) of a filter of length 3 is applied to a univariate time series, by setting the filter values to be equal to $[1/3, 1/3, 1/3]$, the convolution will result in applying a moving average with a sliding window of length 3. The result of a convolution (one filter) on an input time series X can be considered as another univariate time series C that underwent a filtering process. Thus, applying several filters on a time series will result in a multivariate time series whose dimensions are equal to the number of filters used. An intuition behind applying several filters on an input time series would be to learn multiple discriminative features [105]. In this work, a model with one Conv1D layer was adopted.

4.4 Experimental Setup

This chapter is related to **Q2 *How can LSTMs be extended to predict multiple paths?***, by exploring several variants of LSTMs and the way the input data is

[3]The implementation of such layer was obtained from https://pypi.org/project/keras-self-attention/

processed by each of them, it helped to understand better the LSTM architectures and the knowledge obtained contributed to the creation of chapter 5 and chapter 6 where more complex models were used.

As in chapter III, the dataset used for evaluating the models was the KITTI dataset because of its realistic traffic scenes, such as highways, inner city, vehicles standing, vehicle moving, its different objects and the labeled data in image coordinate and 3D information. All this information from the perspective of cameras mouthed on a vehicle which is the main focus of this research. Also, the models were evaluated on three objects – pedestrian, vehicles, and cyclists, for four prediction horizons (P.H.) from ±5 to ±20 steps and only on the BEV perspective, since this is a real measurement of the real world, where the measurements are in meters which represents better the position of the object in the real life than pixels in image coordinates.

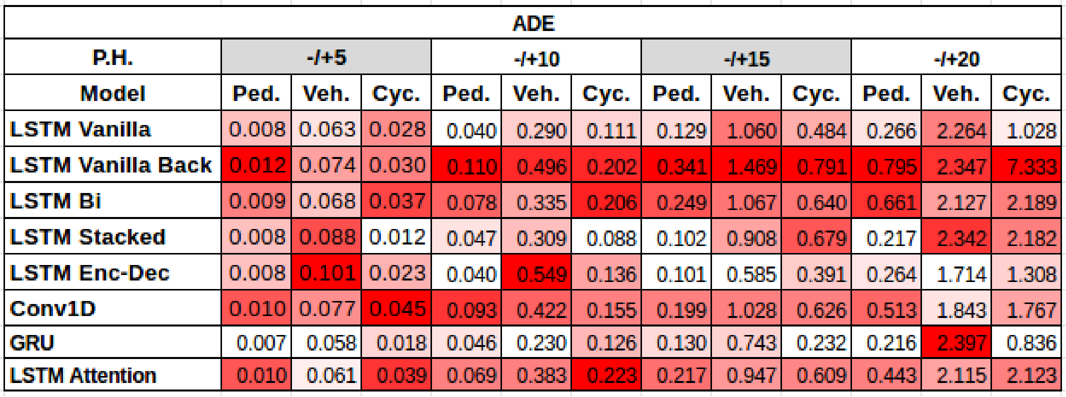
The same data pre-processing steps were used to extract the paths of the KITTI dataset, and the same sliding window method to create the observed tracklets $tr^O = [tr^O_{t1}, tr^O_{t2}, ..., tr^O_{tobs}]$ and its respective ground truth tracklet $tr^G = [tr^G_{t1}, tr^G_{t2}, ..., tr^G_{tpred}]$. The predicted vector of each tr^O is called $tr^P = [tr^P_{t1}, tr^P_{t2}, ..., tr^P_{tpred}]$. As in chapter III, the aim is to predict a tr^P based on the observed tracks tr^O. Also, the same 70%/30% random split was done and the same steps during training and testing were followed.

Finally, Average Displacement Error (ADE) 2.11 and Final Displacement Error (FDE) 2.12 were used to evaluate the performance of each model.

4.5 Results

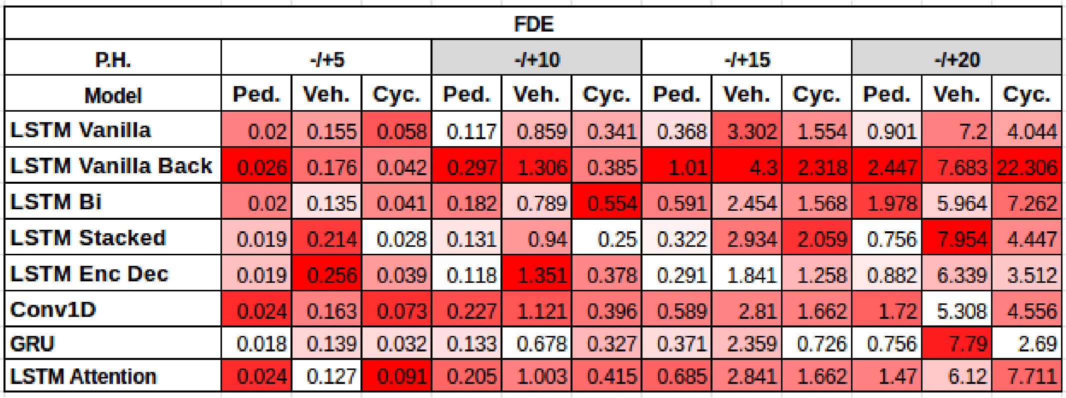
4.5.1 Performance on Each Type of Object

The performance of each model is depicted in Figure 4.3 and Figure 4.4 for the ADE and FDE metrics. The performance is given for four time prediction horizons (P.H.) from ±5 to ±20 and for each object, pedestrian, vehicles, and cyclist. The error is presented in range of color from white to red, where white means that model got the best result (min error) and red means that the model got the worst result (max error). Take for instance the column Ped. for P.H. of ±5 and metric ADE, where the model that got the worst result for the object pedestrian and P.H of ±5 was LSTM Vanilla Backwards (marked in red) and the model that obtained the best result was GRU (marked in white).



ADE												
P.H.	**-/+5**			**-/+10**			**-/+15**			**-/+20**		
Model	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
LSTM Vanilla	0.008	0.063	0.028	0.040	0.290	0.111	0.129	1.060	0.484	0.266	2.264	1.028
LSTM Vanilla Back	0.012	0.074	0.030	0.110	0.496	0.202	0.341	1.469	0.791	0.795	2.347	7.333
LSTM Bi	0.009	0.068	0.037	0.078	0.335	0.206	0.249	1.067	0.640	0.661	2.127	2.189
LSTM Stacked	0.008	0.088	0.012	0.047	0.309	0.088	0.102	0.908	0.679	0.217	2.342	2.182
LSTM Enc-Dec	0.008	0.101	0.023	0.040	0.549	0.136	0.101	0.585	0.391	0.264	1.714	1.308
Conv1D	0.010	0.077	0.045	0.093	0.422	0.155	0.199	1.028	0.626	0.513	1.843	1.767
GRU	0.007	0.058	0.018	0.046	0.230	0.126	0.130	0.743	0.232	0.216	2.397	0.836
LSTM Attention	0.010	0.061	0.039	0.069	0.383	0.223	0.217	0.947	0.609	0.443	2.115	2.123

Figure 4.3: Results ADE. Four P.H. Three Objects.



FDE												
P.H.	**-/+5**			**-/+10**			**-/+15**			**-/+20**		
Model	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
LSTM Vanilla	0.02	0.155	0.058	0.117	0.859	0.341	0.368	3.302	1.554	0.901	7.2	4.044
LSTM Vanilla Back	0.026	0.176	0.042	0.297	1.306	0.385	1.01	4.3	2.318	2.447	7.683	22.306
LSTM Bi	0.02	0.135	0.041	0.182	0.789	0.554	0.591	2.454	1.568	1.978	5.964	7.262
LSTM Stacked	0.019	0.214	0.028	0.131	0.94	0.25	0.322	2.934	2.059	0.756	7.954	4.447
LSTM Enc Dec	0.019	0.256	0.039	0.118	1.351	0.378	0.291	1.841	1.258	0.882	6.339	3.512
Conv1D	0.024	0.163	0.073	0.227	1.121	0.396	0.589	2.81	1.662	1.72	5.308	4.556
GRU	0.018	0.139	0.032	0.133	0.678	0.327	0.371	2.359	0.726	0.756	7.79	2.69
LSTM Attention	0.024	0.127	0.091	0.205	1.003	0.415	0.685	2.841	1.662	1.47	6.12	7.711

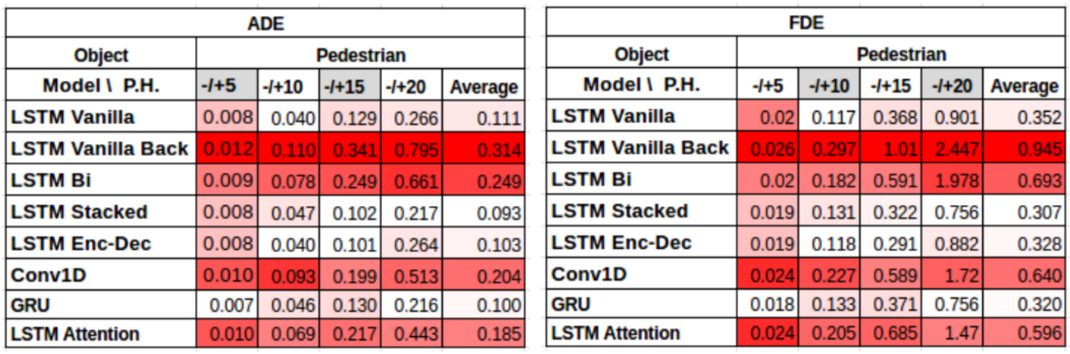
Figure 4.4: Results FDE. Four P.H. Three Objects.

From these results, it can be seen that LSTM Vanilla Backwards shows the worst performance, and the best performance is reached by GRU and LSTM Encoder-Decoder. LSTM Vanilla seems to have have an average performance as in none of the cases is marked with bright red color.

Looking a bit closer by time prediction horizon, GRU seems to keep good performance along the whole P.H., LSTM Encoder-Decoder appears to struggle with short P.H., from ± 5 to ± 10, but its performance improves for long P.H., from ± 15 to ± 20. Conv1D shows similar behaviour that LSTM Encoder-Decoder. LSTM Attention, also tends to get better performance for long P.H., from ± 15 to ± 20; this goes with the nature of attention models, since they were created to deal with long sequences. LSTM Bi does not present any specific pattern in its behaviour. LSTM Stacked and LSTM Vanilla, both show an average performance performance across P.H. Finally LSTM Vanilla Backwards shows to have better performance for short P.H of ± 5.

The results ordered by type of object and time prediction horizon is depicted in Figure 4.5, Figure 4.6 and Figure 4.7. For the object pedestrian in Figure 4.5, the best models are GRU, LSTM Encoder-Decoder, and LSTM Stacked. LSTM Vanilla still shows an average performance, followed by Conv1D and LSTM Attention. For the object vehicle, as depicted in Figure 4.6, it can be seen that GRU reach good results for P.H. from ± 5 to ± 15, and for long P.H. from ± 15 to ± 20, LSTM Encoder-Decoder got good results. LSTM Vanilla and LSTM Attention keeps an average performance similar to LSTM Bidirectional. LSTM Stacked and LSTM Vanilla Backwards show the worst performance. Finally for the object Cyclist in Figure 4.6, the best model is GRU. LSTM Stacked got the best results but it struggles for long P.H. from ± 15 to ± 20. LSTM Vanilla, LSTM Encoder-Decoder, Conv1D and LSTM Attention present an average performance along with LSTM Bi. LSTM Vanilla Back shows the worst performance.

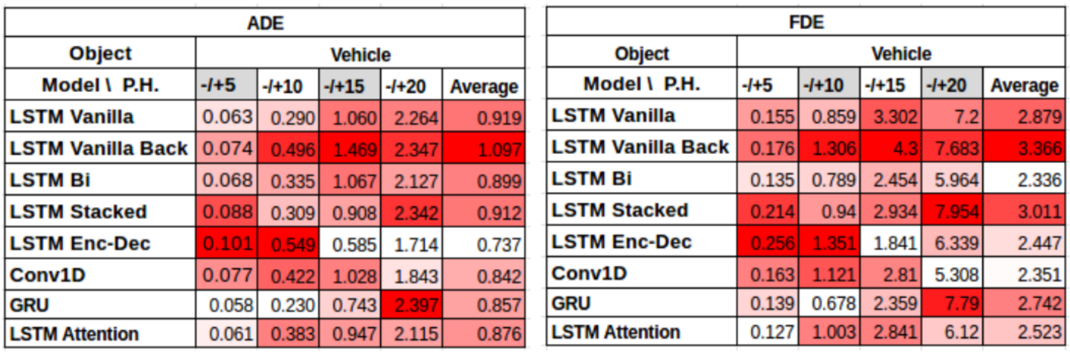
Considering the type of object, all the models appear to struggle when predicting trajectories for the object vehicle, followed by the object cyclist and for the object pedestrian, as shown in Figure 4.8 and Figure 4.9. From these figures it can be seen as well that performance of all the models decreases when they have to predict for long time prediction horizons.



ADE					
Object	Pedestrian				
Model \ P.H.	-/+5	-/+10	-/+15	-/+20	Average
LSTM Vanilla	0.008	0.040	0.129	0.266	0.111
LSTM Vanilla Back	0.012	0.110	0.341	0.795	0.314
LSTM Bi	0.009	0.078	0.249	0.661	0.249
LSTM Stacked	0.008	0.047	0.102	0.217	0.093
LSTM Enc-Dec	0.008	0.040	0.101	0.264	0.103
Conv1D	0.010	0.093	0.199	0.513	0.204
GRU	0.007	0.046	0.130	0.216	0.100
LSTM Attention	0.010	0.069	0.217	0.443	0.185

FDE					
Object	Pedestrian				
Model \ P.H.	-/+5	-/+10	-/+15	-/+20	Average
LSTM Vanilla	0.02	0.117	0.368	0.901	0.352
LSTM Vanilla Back	0.026	0.297	1.01	2.447	0.945
LSTM Bi	0.02	0.182	0.591	1.978	0.693
LSTM Stacked	0.019	0.131	0.322	0.756	0.307
LSTM Enc-Dec	0.019	0.118	0.291	0.882	0.328
Conv1D	0.024	0.227	0.589	1.72	0.640
GRU	0.018	0.133	0.371	0.756	0.320
LSTM Attention	0.024	0.205	0.685	1.47	0.596

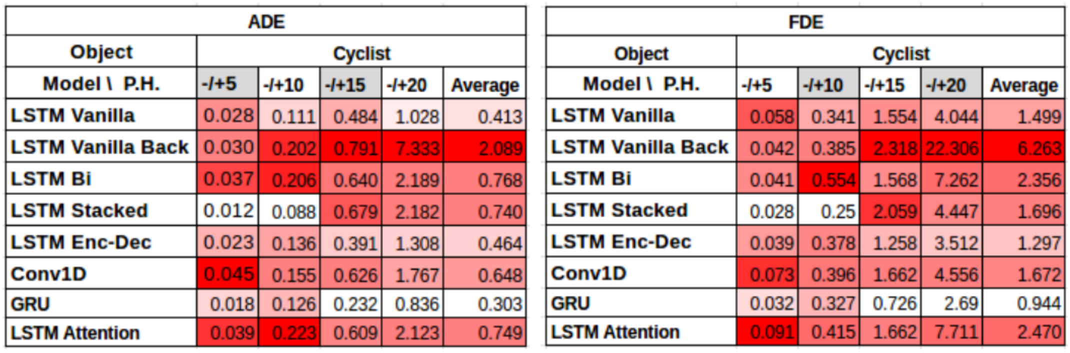
Figure 4.5: Results ADE and FDE. Four P.H. Pedestrians.



ADE					
Object	Vehicle				
Model \ P.H.	-/+5	-/+10	-/+15	-/+20	Average
LSTM Vanilla	0.063	0.290	1.060	2.264	0.919
LSTM Vanilla Back	0.074	0.496	1.469	2.347	1.097
LSTM Bi	0.068	0.335	1.067	2.127	0.899
LSTM Stacked	0.088	0.309	0.908	2.342	0.912
LSTM Enc-Dec	0.101	0.549	0.585	1.714	0.737
Conv1D	0.077	0.422	1.028	1.843	0.842
GRU	0.058	0.230	0.743	2.397	0.857
LSTM Attention	0.061	0.383	0.947	2.115	0.876

FDE					
Object	Vehicle				
Model \ P.H.	-/+5	-/+10	-/+15	-/+20	Average
LSTM Vanilla	0.155	0.859	3.302	7.2	2.879
LSTM Vanilla Back	0.176	1.306	4.3	7.683	3.366
LSTM Bi	0.135	0.789	2.454	5.964	2.336
LSTM Stacked	0.214	0.94	2.934	7.954	3.011
LSTM Enc-Dec	0.256	1.351	1.841	6.339	2.447
Conv1D	0.163	1.121	2.81	5.308	2.351
GRU	0.139	0.678	2.359	7.79	2.742
LSTM Attention	0.127	1.003	2.841	6.12	2.523

Figure 4.6: Results ADE and FDE. Four P.H. Vehicles.



ADE					
Object	Cyclist				
Model \ P.H.	-/+5	-/+10	-/+15	-/+20	Average
LSTM Vanilla	0.028	0.111	0.484	1.028	0.413
LSTM Vanilla Back	0.030	0.202	0.791	7.333	2.089
LSTM Bi	0.037	0.206	0.640	2.189	0.768
LSTM Stacked	0.012	0.088	0.679	2.182	0.740
LSTM Enc-Dec	0.023	0.136	0.391	1.308	0.464
Conv1D	0.045	0.155	0.626	1.767	0.648
GRU	0.018	0.126	0.232	0.836	0.303
LSTM Attention	0.039	0.223	0.609	2.123	0.749

FDE					
Object	Cyclist				
Model \ P.H.	-/+5	-/+10	-/+15	-/+20	Average
LSTM Vanilla	0.058	0.341	1.554	4.044	1.499
LSTM Vanilla Back	0.042	0.385	2.318	22.306	6.263
LSTM Bi	0.041	0.554	1.568	7.262	2.356
LSTM Stacked	0.028	0.25	2.059	4.447	1.696
LSTM Enc-Dec	0.039	0.378	1.258	3.512	1.297
Conv1D	0.073	0.396	1.662	4.556	1.672
GRU	0.032	0.327	0.726	2.69	0.944
LSTM Attention	0.091	0.415	1.662	7.711	2.470

Figure 4.7: Results ADE and FDE. Four P.H. Cyclists.

4.6 Discussion and Conclusions

The objective if this chapter was to compare the performance of some models against the LSTM Vanilla model on predicting the future position of objects commonly presents in traffic scenes. The models were tested on predicting in a birds' eye view map. The following conclusions can be drawn:

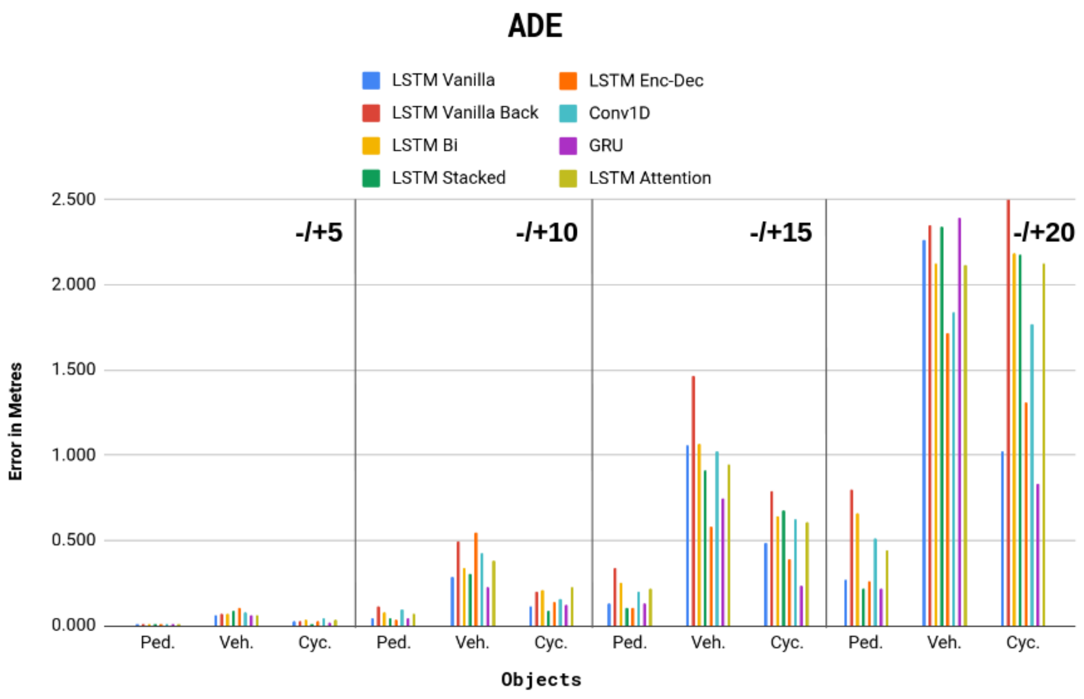


Figure 4.8: Results ADE. Four P.H. Three Objects.

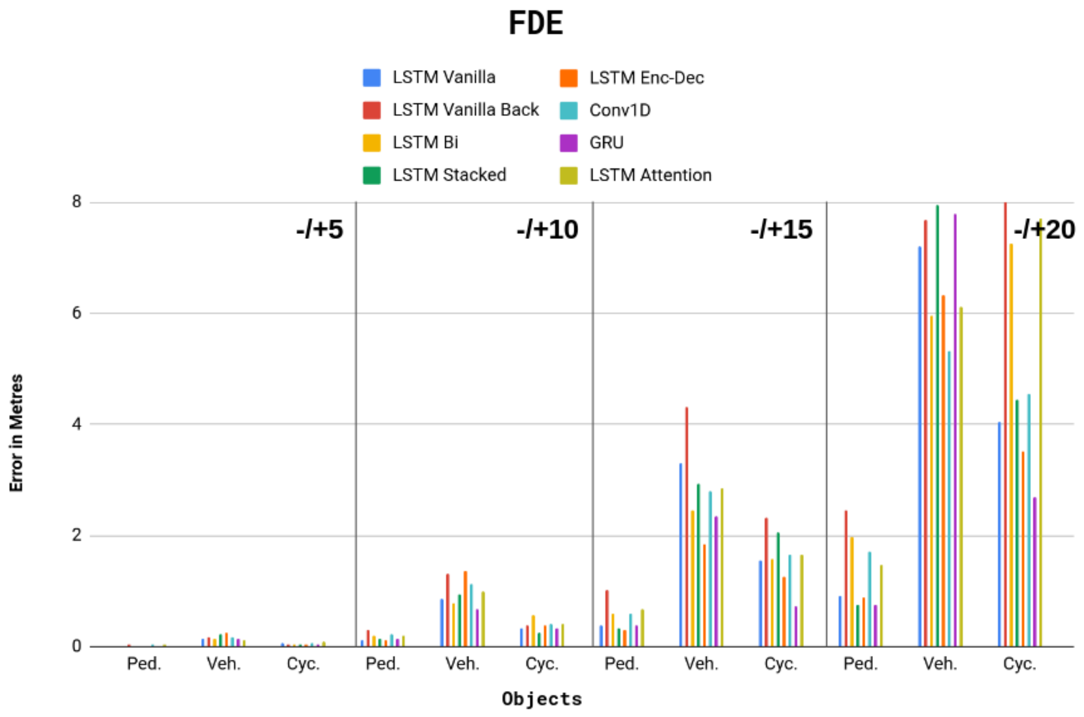


Figure 4.9: Results FDE. Four P.H. Three Objects.

1. LSTM Vanilla have an average performance compared with the other models.
2. The performance of all the models decrease when predicting for long time prediction horizons (P.H).
3. None of the model is the best in all the cases.
4. GRU is the model that performs better in most of the cases.
5. Some models are better for predicting long time P.H. such as LSTM Encoder-Decoder and ConV1D which would be interesting to analyse in further work.
6. Different to the studies shown in the introduction, in this experiments the improvement in performance of some models against the vanilla is significant and this significance increase when longer the time prediction horizon is. However, it is important to say that non of the models perform better than the vanilla LSTM in all the cases.
7. Considering the type of object, all the models appear to struggle when predicting trajectories for the object vehicle, followed by the object cyclist and for the object pedestrian.
8. Table 4.2 and Table 4.3 present the average performance of the models for the metrics ADE and FDE respectively. The average performance is by object type on the four time P.H. and ordered in a ascending manner. Remember, small error values means that the model performs better.
9. Finally, this chapter allows us to understand better how to use LSTM architectures in its different ways, since this will be needed it in the next chapters where more complex model are designed for predicting multiple paths and for processing multimodal data.

Table 4.2: Performance of the models by average in each of the objects. Metric: ADE

ADE					
Pedestrian		**Vehicles**		**Cyclists**	
Model	**Average**	**Model**	**Average**	**Model**	**Average**
LSTM Stacked	0.093	LSTM Enc-Dec	0.737	GRU	0.303
GRU	0.100	Conv1D	0.842	LSTM Vanilla	0.413
LSTM Enc-Dec	0.103	GRU	0.857	LSTM Enc-Dec	0.464
LSTM Vanilla	0.111	LSTM Attention	0.876	Conv1D	0.648
LSTM Attention	0.185	LSTM Bi	0.899	LSTM Stacked	0.740
Conv1D	0.204	LSTM Stacked	0.912	LSTM Attention	0.749
LSTM Bi	0.249	LSTM Vanilla	0.919	LSTM Bi	0.768
LSTM Vanilla Back	0.314	LSTM_Vanilla_Back	1.097	LSTM Vanilla Back	2.089

Table 4.3: Performance of the models by average in each of the objects. Metric: FDE.

FDE					
Pedestrian		**Vehicles**		**Cyclists**	
Model	**Average**	**Model**	**Average**	**Model**	**Average**
LSTM Stacked	0.307	LSTM Bi	2.336	GRU	0.944
GRU	0.320	Conv1D	2.351	LSTM Enc-Dec	1.297
LSTM Enc-Dec	0.328	LSTM Enc-Dec	2.447	LSTM Vanilla	1.499
LSTM Vanilla	0.352	LSTM Attention	2.523	Conv1D	1.672
LSTM Attention	0.596	GRU	2.742	LSTM Stacked	1.696
Conv1D	0.640	LSTM Vanilla	2.879	LSTM Bi	2.356
LSTM Bi	0.693	LSTM Stacked	3.011	LSTM Attention	2.470
LSTM Vanilla Back	0.945	LSTM Vanilla Back	3.366	LSTM Vanilla Back	6.263

Chapter 5

LSTMs and MDN for Multiple Path Prediction

5.1 Introduction

As established in the previous chapters, path prediction using RNNs has shown good performance. However, most of the approaches have the limitation of only predicting a single path per tracklet. Path prediction is not a deterministic task and requires predicting with a level of uncertainty. In addition, generating a set of paths instead of a single one is a more realistic manner of predicting the possible position of objects. Some works only focus on specific scenarios such as intersections, crossing roads and highways from a top view where the movements of the objects are limited by the shape of the scenarios. Nevertheless, real-life traffic scenarios are more diverse and consequently the movements of the objects in that environment are also diverse. This chapter focus on extending LSTM architectures to predict multiple paths per observed tracklet along with with associated uncertainty. The approach is evaluated on two datasets, KITTI that gives annotated data from cameras mouthed on a vehicle and CityFlow that provide data from surveillance cameras. Both datasets content realistic and diverse scenarios from traffic scenes.

This chapter is mainly related to RQ2 since here MDN (Mixture Density Networks) are used to extend LSTM architecture to predict a set of paths instead of a single one.

This chapter contains partial content published (peer-reviewed) in the VEHITS

2020 conference. In this chapter, Section 5.2 describes the data and the aim of this chapter. Section 5.3 presents our approach. Section 5.4 and 5.5 present the experimental setup and results respectively. Finally, in section 5.6 a discussion and conclusions are given.

5.2 Data Definition

In this work we apply a sliding window over one track per time period then these smaller segments are split into two vectors of equal size. The first vector is the observed tracklets $tr^O = [tr^O_{t1}, tr^O_{t2}, ..., tr^O_{tobs}]$ and the second vector is its respective ground truth tracklet $tr^G = [tr^G_{t1}, tr^G_{t2}, ..., tr^G_{tpred}]$. The predicted vector of each tr^O is called $tr^P = [tr^P_{t1}, tr^P_{t2}, ..., tr^P_{tpred}]$. We aim to predict tr^P based on the observed tracks tr^O but instead of only predicting one tr^P we want to predict a set, tr^{PS}, of m tr^P per each tr^O with its respective probability such that $tr^{PS} = [tr^P_1, tr^P_2, ..., tr^P_m]$ and $tr^P_x = [tr^P, Probability]$, as illustrated in Figure 5.1.

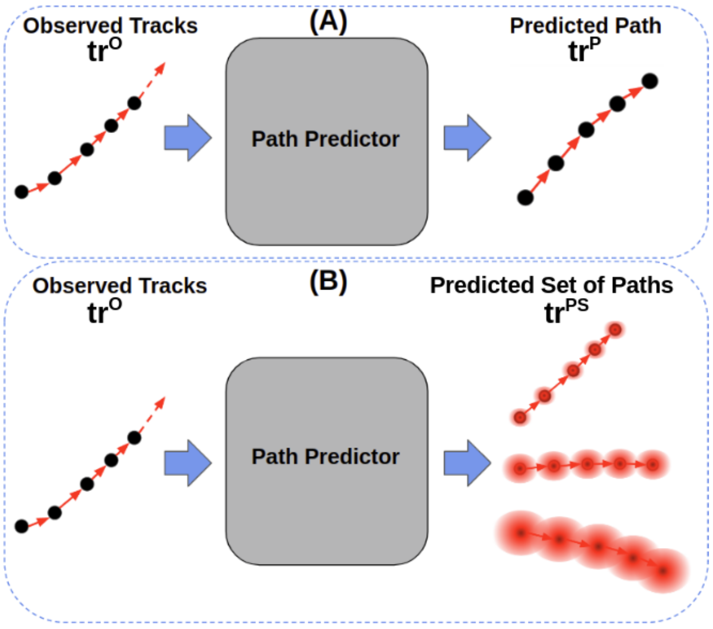


Figure 5.1: Predicting one path (A) vs predicting a set of paths (B).

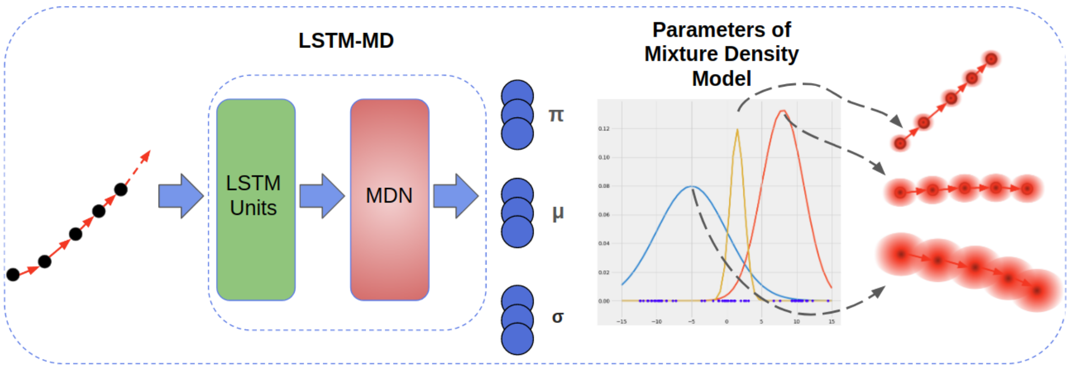


Figure 5.2: General Proposed Approach.

5.3 Approach

As discussed in Chapter 4, LSTMs have previously shown good performance when dealing with sequential or time series data, so in this approach an LSTM architecture is used. LSTMs can be used in different ways, one of which is the Multiple Output Strategy (MOS). MOS develops one model to predict an entire sequence in a one-shot manner, outputting a vector directly that can be interpreted as a multi-step forecast. However, at this stage the problem of only being able to predict a single path per observed tracklet still remains. To overcome this limitation, we use the well known properties of Mixture Density Models (MDMs) and inspired by [2], we propose to use LSTMs with MDMs as a MDN layer, as shown in Figure 5.2.

5.3.1 Model Architecture

The core of the model are two stacked LSTMs with a final MDN layer. The number of inputs and outputs depends on the length of the observed tracklet, tr^O, and the number of steps to be predicted ahead, tr^P. The Keras API [1] and the Keras MDN Layer library [2] were used to implement the LSTM architecture and the MDN Layer respectively. Figure 5.3 shows the architecture used.

[1]https://keras.io/
[2]https://pypi.org/project/keras-mdn-layer/

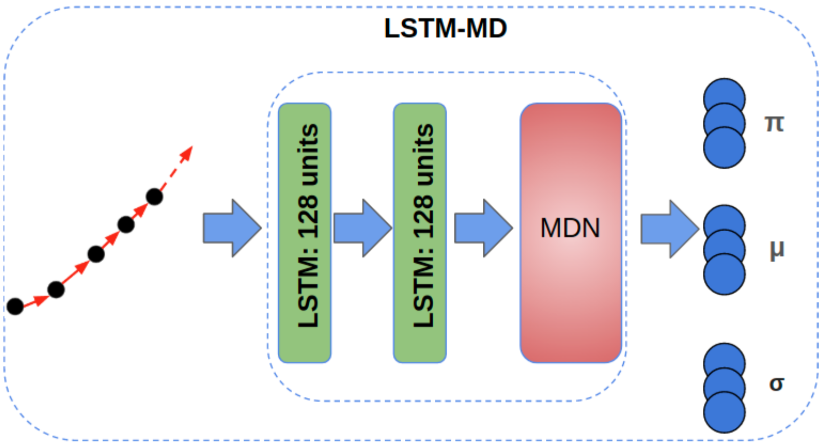


Figure 5.3: LSTM and MDN Architecture.

5.3.2 Training

During training, from the available data a random training and test split was done; 70% of the data was used for training and the rest for testing. Also, to feed the model correctly, the next steps were followed:

1. Specify the length of tracklets to be processed and the number of mixtures to be used, $NMixes$. The number of mixtures $NMixes$ is the number of paths to be predicted per observed tracklet, tr^O. Experiments using 2, 3, 4 and 5 $NMixes$ were performed.

2. Scale data [0,1]: un-scaled input variables can result in a slow or unstable learning process, whereas un-scaled target variables on regression problems can result in exploding gradients causing the learning process to fail.

3. Split data into observed tracklet, tr^O, and ground truth path (tracklet) to be predicted, tr^G.

4. tr^O is shaped as $[Nsamples, tobs, features\text{-}in\text{-}tr^O]$ and tr^G as $[Nsamples, OutputLength]$. Where $Nsamples$ is the number of samples in the dataset, $tobs$ is the number of tracks in each observed tracklet (tr^O), $features\text{-}in\text{-}tr^O$ is the number of features in each observed track. Finally, $OutputLength$ is

the size of the output of the model. In this case, $OutputLength = tpred * features\text{-}in\text{-}tr^G$. Here $tpred$ is the steps to predict in the future and $features\text{-}in\text{-}tr^G$ is the number of features to predict in each step.

5.3.3 Multiple Trajectory Extraction

As mentioned before, during training we have to specify the number of paths to be predicted per observed tracklet tr^O by setting $NMixes$. In this work experiments using 2, 3, 4 and 5 $NMixes$ were performed to analyse the behaviour of the model. During testing it is not necessary to specify how many path we want since the model have been trained to predict $NMixes$ or N paths per observed tracklet tr^O.

During prediction, instead of having as output a single array of $[tpred, x, y]$ positions, the model outputs an array with the parameters of a Mixture Density Model, which are mean, standard deviation (SD) and mixing coefficients or proportions (Mp). These parameters are given in the format of $[Mean_1, Mean_2, ..., Mean_N, SD_1,\ \ SD2, ..., SD_N, Mp_1, Mp_2, ..., Mp_N]$. Where $Mean_N$ is a set of $[tpred, x, y]$ positions of each predicted paths. SD_N is a set of $[tpred, SD_x, SD_y]$ values for each $Mean_N$. Finally, Mp_N contains per column an individual value per each $Mean_N$. All this is given as a flattened array. An example of the flattened array output by the model when predicting 1 step ahead and 5 possible paths is given in Figure 5.4.

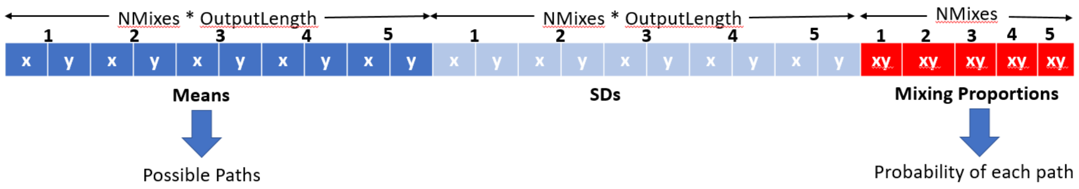


Figure 5.4: Multiple trajectory extraction: example predicting 1 step ahead and 5 possible paths. $OutputLength = 1 * 2$. 1 step predicted ahead and 2 features (x,y). $Nmixes = 5$ possible paths.

Regarding the mentioned array and taking into account the structure of the data in it, the output of the model was processed as follow to extract the multiple trajectories:

1. Extracting mean: the first $NMixes * OutputLength$ columns are extracted as means and the array is resized to an array of $Mean_{PredictedPaths} = [NMixes, tpred, featuresintr^G]$ which can be read as Path number, steps to predict in the future, the number of features to predict in each step (x, y).

2. Extracting standard deviation (SD): the second $NMixes * OutputLength$ columns are extracted as SDs and similar to the means the array is resized to an array of $SDs_{PredictedPaths} = [NMixes, tpred, featuresintr^G]$ which can be read as Path number, steps to predict in the future, the number of features to predict in each step (x, y).

3. Extracting mixing proportions (Mp): the last $NMixes$ columns are the Mp of each predicted path and are stored in an array $Mp_{PredictedPaths}$.

4. Finally, once each array is set up, each row in $Mean_{PredictedPaths}$ is considered as a possible path and their mixing proportion, stored as columns in $Mp_{PredictedPaths}$, are the probabilities of each path. $SDs_{PredictedPaths}$ was not used in this experiments as the standard deviation was not needed for further analysis of the results or for visualization purpose as it was the case of the means and the Mps.

5.4 Experimental Setup

A second question to answer in this research work is: **Q2** ***How can LSTMs be extended to predict multiple paths?***. As explained above LSTMs are used along MDN (Mixture Density Networks) to predict a set of paths instead of a single one. Also, an initial exploration of **Q3** ***Are Long Short-Term Memory (LSTM) architectures suitable for sequential and enriched trajectory information?***

is performed by adding 3 features more to the positional information of the tracklets (LMDN5). These experiments are related to explore these two research questions.

The datasets used for evaluating the models is KITTI and CityFlow. KITTI because of its realistic scenes, such as highways, inner city, vehicles standing, vehicle moving, its different objects and the labeled data in image coordinate and 3D information. All this from cameras mouthed on a vehicle which is the main focus of this research work. CityFlow is used in this chapter as well because it is a dataset that contains annotated data of images from static surveillance cameras from traffic scenes. The dataset covers a diverse set of location types, including intersections, stretches of roadways, and highways. The surveillance perspective can be seen as a birds-eye view of the objects and it is perfect to visualize the predicted paths gotten from the multiple path prediction approach. Another reason to use CityFlow in this chapter is that at the time of exploring LSTMs and MDN, we were using this dataset to perform other research task called object re-identification so we got familiarized with this dataset and we already know which information could be used and how to used, so its use did not implicated any more analysis on understandings the data. However, it was not used in the following chapters because it only provides positional information of the object vehicle from static cameras in image coordinates (pixels), besides it does not provide other type of information such as ego motion as KITTI does.

As in chapter 3, the models were evaluated on three objects – pedestrian, vehicles, and cyclists for KITTI and vehicles for CityFlow. Four prediction horizons (P.H.) from ± 5 to ± 20 steps were used.

The same data pre-processing steps were used to extract the trajectories from the datasets, and the same sliding window method was used to create the observed tracklets $tr^O = [tr^O_{t1}, tr^O_{t2}, ..., tr^O_{tobs}]$ and its respective ground truth tracklet $tr^G = [tr^G_{t1}, tr^G_{t2}, ..., tr^G_{tpred}]$. Finally, Average Displacement Error (ADE) 2.11 and Final Displacement Error (FDE) 2.12 were used to evaluate the performance of each model.

Table 5.1 summarizes the number of tracklets extracted in each dataset. 70% were used for training and the rest for testing:

Table 5.1: Size of data for all P.H. and objects.

Number of Tracklets				
	KITTI			**CityFlow**
P.H.	**Pedestrian**	**Vehicle**	**Cyclist**	**Vehicle**
$\pm$ 5	9,992	26,112	1,605	37,480
$\pm$10	8,551	20,454	1,285	31,164
$\pm$15	7,330	16,031	1,028	25,936
$\pm$20	6,312	13,027	802	21,923

The comparative study was performed with two baselines methodologies – Kalman Filter (KF) with Constant Velocity (CV) model and Vanilla LSTM (VLSTM) with 128 neurons – to establish that our approach does not lose accuracy when predicting a set of paths.

For our proposed model (LSTM with MDM), we performed two experiments for image coordinates. Instead of only using two features, (x, y) position of objects (LMDN2), we included three additional features: height (h), object area $(objectA)$ and object area with respect to the image $(objectAImg)$. This produced a feature vector of $(x, y, h, objectA, objectAImg)$ (LMDN5).

5.5 Results

The performance was calculated on four different prediction horizons (P.H.) and for three different objects – pedestrians, cyclists and vehicles (Cars, Vans or Trucks). The results are also provided in image coordinates (pixels) and in birds-eye view (metres). Due to the size of the images, the results show large numerical values in the case of image coordinates. Finally, the approach was also evaluated on the generation of different numbers of paths, two to five. The results (Tables 5.2, 5.3 & 5.4) show the accuracy of the mixture component with the highest probability compared with the baseline methods. Examples of predicting different numbers of paths are shown in the visualizations (Fig. 5.5 & 5.6). Where it is depicted how the predicted paths differ gradually from the ground truth according to the given probability. It is also shown that predicting up to three paths per observed tracklet still gives consistent paths.

5.5.1 Performance on the KITTI Dataset

Table 5.2 shows the performance of the approach for image coordinates. As expected, all methods show the error increases directly with the predicted horizon: the larger the P.H., the larger the error. Table 5.2 shows that our approach, LMDN2, achieves better accuracy than the two baseline methods. It can also be observed that the approach LMDN5 showed improvement for short P.H. in ADE and in several cases for FDE, mostly for the objects pedestrian and vehicle.

Table 5.3 presents the results calculated using birds-eye view (BEV). For most cases, our approach, LMDN2, achieves better accuracy than the two baseline methods. An exception is a significant error increase in the case of the vehicle class for the P.H. of ± 10 for both ADE and FDE.

Figure 5.5 shows the resulting set of paths predicted when configuring the model to have five mixtures. The first 1,000 samples are displayed. The first image (GT) displays the ground truth paths and the following images from MDN 1 to MDN 5 present the paths predicted by each mixture component. The mixtures were ordered according to descending probability of their components – MDN 1 (high probability) to MDN 5 (low probability). The predicted paths diverge from the ground truth when the probability of the the components used for predicting go from high to low.

5.5.2 Performance on the CityFlow Dataset

Table 5.4 presents the performance of the methods on the CityFlow dataset. For all approaches, similar behaviour to that in KITTI can be observed – the error increases directly with the P.H. Table 5.4, shows that our approach, LMDN2, achieves better accuracy than the two baseline methods. The approach LMDN5 (using 5 features) showed significant improvement over the method LMDN2 for the P.H. of ± 5, ± 10,

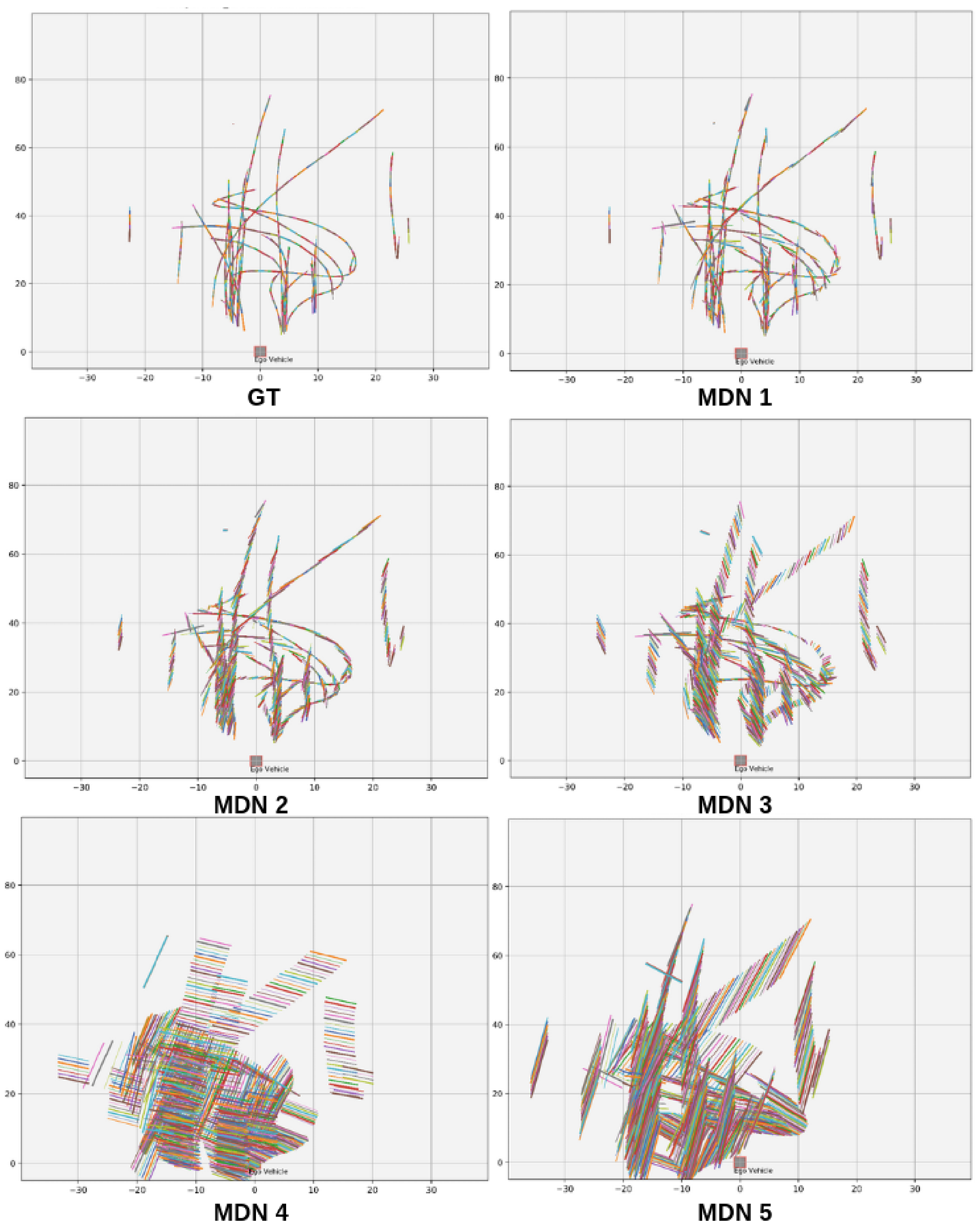


Figure 5.5: Set of paths predicted using five mixtures. Dataset: KITTI. P.H.:±5. Point of view: BEV.

Table 5.2: Path prediction accuracy on KITTI. Image coordinates.

KITTI. Image Coordinate (Pixels)												
Method	**LMDN5**			**LMDN2**			**VLSTM**			**KF**		
	ADE			**ADE**			**ADE**			**ADE**		
P. H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	79	65	232	106	84	107	76	98	94	111	225	143
±10	286	272	329	362	267	261	244	353	160	260	585	287
±15	466	591	1596	660	614	467	802	668	1163	549	1019	807
±20	1407	1035	15673	1185	723	2854	1755	1065	6856	970	1554	1944
	FDE			**FDE**			**FDE**			**FDE**		
P. H.	**Ped.**	**Veh.**	**Cyc.**	**Pe.**	**Veh.**	**Cyc**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	168	161	502	237	220	277	169	252	271	214	501	385
±10	721	896	1124	1098	924	946	727	1163	538	778	1914	1022
±15	1517	2265	3742	2231	2230	1917	2483	2412	4452	1907	3695	3408
±20	5061	3678	38492	4100	2828	12346	6251	4040	25743	3567	5808	8539

and ±15 for both ADE and FDE.

To predict a set of paths, from 2 to 5, similar behaviour to that in the KITTI dataset is observed. The predicted paths diverge from the ground truth when the probability of the components used for predicting goes from high to low. Figure 5.6 depicts one example of predicting from 2 to 5 sets of paths for a P.H. of ±5. When the predicted paths are near the ground truth, the probability of such a path is high, in contrast, when the predicted paths are far from the ground truth, their probability is low. In some cases, as in figure 5.6 at row 3 and 4 left, the paths are not displayed because the probability of that path is too low.

Table 5.3: Path prediction accuracy on KITTI. BEV.

KITTI. Bird-Eye View (Meters)									
Method	**LMDN2**			**VLSTM**			**KF**		
	ADE			**ADE**			**ADE**		
P. H.	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	0.01	0.06	0.02	0.01	0.06	0.02	0.06	0.47	0.24
±10	0.04	0.55	0.11	0.05	0.25	0.10	0.10	0.75	0.36
±15	0.10	0.73	0.285	0.12	0.87	0.41	0.21	1.12	0.54
±20	0.31	1.31	0.805	0.22	1.68	1.01	0.41	1.79	0.90
	FDE			**FDE**			**FDE**		
P. H.	**Ped.**	**Veh.**	**Cyclist**	**Ped.**	**Veh.**	**Cyc.**	**Ped.**	**Veh.**	**Cyc.**
±5	0.02	0.13	0.03	0.02	0.15	0.04	0.08	0.58	0.27
±10	0.13	1.24	0.29	0.14	0.72	0.27	0.25	1.67	0.64
±15	0.32	2.16	1.08	0.38	2.69	1.29	0.69	3.28	1.42
±20	1.13	4.21	2.21	0.77	5.48	3.06	1.49	6.04	2.95

Table 5.4: Path prediction accuracy on CityFlow. Image coordinates.

CityFlow. Image Coordinate (Pixels).				
Method	**LMDN5**	**LMDN2**	**VLSTM**	**KF**
P.H.	**ADE**	**ADE**	**ADE**	**ADE**
±5	486	582	634	1369
±10	665	905	893	1948
±15	891	1094	1134	1846
±20	1613	1371	1587	2202
P.H.	**FDE**	**FDE**	**FDE**	**FDE**
±5	878	1125	1209	2492
±10	1792	2659	2525	5167
±15	3084	3817	4135	6585
±20	5749	5197	6121	8382

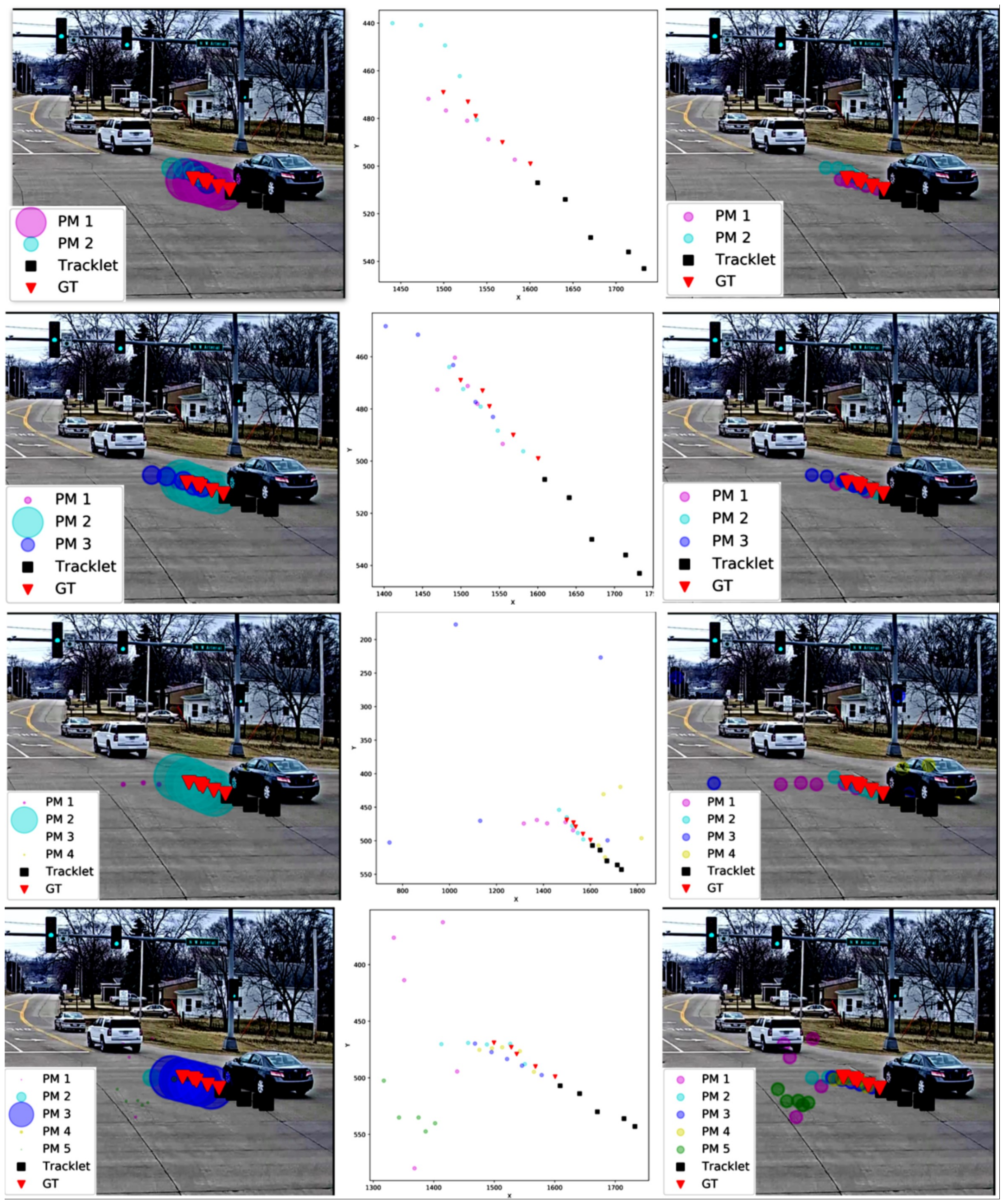


Figure 5.6: From top to bottom, predicting two (row 1), three (row 2), four (row 3) and five (row 4) sets of paths. Left: shows the predicted set of paths with their respective probability (the larger the circle, the larger the probability of that path). Middle and right: present a close-up of the set of paths without their probabilities. Dataset: CityFlow. P.H.:±5. Point of view: Image Coordinate.

5.6 Discussion and Conclusions

This chapter presents an approach for predicting multiple paths with associated uncertainty for forecasting near future positions of objects commonly present in traffic scenes. The objective of this work was to explore the performance of the combination of LSTM and MDN architectures and analyze the parameters output by these models for predicting a set of paths. The evaluation was made for three object classes, four P.H. and two different points of view.

The comparative study shows that LSTMs do not decrease their performance when combined with the MDN. Regarding the method LMDN5, the experiments showed that the extra features lead to better results overall. This was evidenced in KITTI for short prediction horizons for the object pedestrian and vehicle and for CityFlow for the object vehicle for P.H. of less than ± 15. The relationship between the accuracy of the predicted set of paths and the probability of each component in the MDN model can be seen in Figure 5.5 and Figure 5.6. As expected, the predicted paths are more similar to the ground truth when the component that is predicting them gives high probability. However, those paths that are being predicted for the components with low probability are increasingly different to ground truth. This conclusion is desirable when predicting paths, because based on the probability of each predicted path we can relay more in those paths with high probability.

The approach was also evaluated for predicting multiple numbers of paths per input tracklet. It was observed that when predicting two to three paths per input, the approach works well as the predicted paths are still related to the ground truth. However, in some cases, when predicting four and five paths, some of the predicted paths begin to deviate further from the ground truth. This cannot be seen as a disadvantage since each path has a probability, so by looking at the probability of each path, those paths with very low probability can be discarded.

The approach performs better when predicting in birds-eye view than when predicting in image coordinates. However, such 3D information is not always available. The reason for this could be that using pixels is not the best way of representing the position of the object in an image since is too sensitive to small movements of the camera and also of the object detection. Something to consider here is that when predicting in image coordinates, further normalisation is needed to counter the size of the images in the dataset. Results in pixels were provided here to be consistent with other published results. As shown in our experiments, the errors in pixels are large but that does not mean that the predictions are far from the ground truth paths.

The results have shown that the use of LSTM with a Mixture Density Model achieves good performance of up to an ADE of 0.01m for pedestrians, 0.06m for vehicles and 0.02m for cyclists and up to an FDE of 0.02m, 0.13m, 0.03m for the same objects using BEV and P.H. of ± 5. The results also show that the performance is affected by the P.H. where longer horizons result in a larger displacement error. The P.H. where the approach is more reliable is for ± 5 and ± 10 for image coordinate and up to ± 15 for birds-eye view. The FPS in both datasets is 10 therefore we are predicting from ($\pm$ 0.5s) to ($\pm$2s) seconds ahead.

Finally, the processing inference time per tracklet for our approach was 0.044ms/tr, 0.055ms/tr, 0.084ms/tr, 0.102ms/tr for P.H. of ± 5 to ± 20 respectively. This was measured using a PC with the following features: GPU GeForce GTX 980, CPU Intel® Core^{TM} i5-4690K CPU @ 3.50GHz x 4, RAM 24GB

Chapter 6

Path Prediction Using Contextual Information

6.1 Introduction

In previous chapters only the past position of the observed object in a scene have been used to predict its future path. However, in traffic scenarios there is a rich set of additional information available about the environment of the ego vehicle and each object in the scene. For example, this information could be an image of a moving object, the velocity of the ego vehicle, the position of other objects or an image of the scene itself. Nowadays, instrumented vehicles are capable of sensing and providing this information that could be leveraged in the path prediction task. This information (contextual information or cues) are used along with the $tr(x, y)$ positional information of the object whose future path is to be predicted [101, 110, 98, 81].

However, using this contextual information still poses a challenge. Since this information comes in different types of data, including numerical and image data, and from different sources, several problems have to be faced such as synchronization and availability of data, feature extraction, multimodal data management, and data fusion strategies. This chapter presents the approach developed in this research thesis to use this contextual information in the path prediction task.

This chapter is related to RQ1, RQ3 and RQ4 since this chapter explore the representation of positional data in a latent space (RQ1) and the processing of context information (RQ3) along with its different representations and fusion strategies (RQ4).

For the remainder of this chapter, Section 6.2 introduces the contextual information used. Section 6.3 defines the nature of the data. Section 6.4 presents our approach. Section 6.5 describes the experimental setup. Section 6.6 presents an initial exploration of the ego-vehicle features along with the obtained results. Followed by section 6.7 where a deeper exploration of multimodal features is detailed. Section 6.8 describes the use of ensembles to improve the performance of individual models. Finally, in Section 6.9 conclusions are given.

6.2 Contextual Information

For traffic scenes, and specifically from cameras mounted on a moving vehicle (ego vehicle), the following cues or features were used: object, ego-vehicle telemetery data, scene, and interaction aware or social context features (other objects), as visualised in Figure 6.1. These four type of features were selected since they cover most of the information that can be obtained from the perspective of an ego vehicle. More details about these features are given below.

6.2.1 Object

Besides the x, y position of the object, regions of the objects from the RGB images are used. These images can potentially give better understanding of the object such as the pose or orientation that can be leveraged as additional features, Figure 6.2.

6.2.2 Ego vehicle

Some data sets provide the dynamics of the vehicle used to record the scene. KITTI provides telemetry of the ego vehicle, such as orientation, velocity in [x,y,z] and acceleration in [x,y,z]. The configuration of the axes are shown in Figure 6.3.

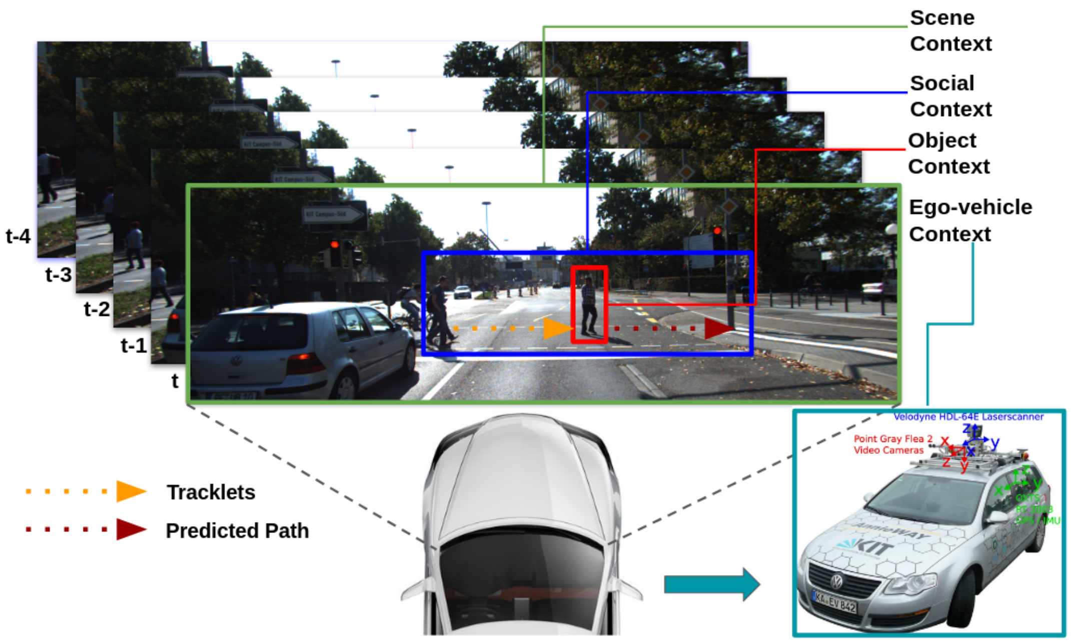


Figure 6.1: Contextual information from Ego-vehicle view.

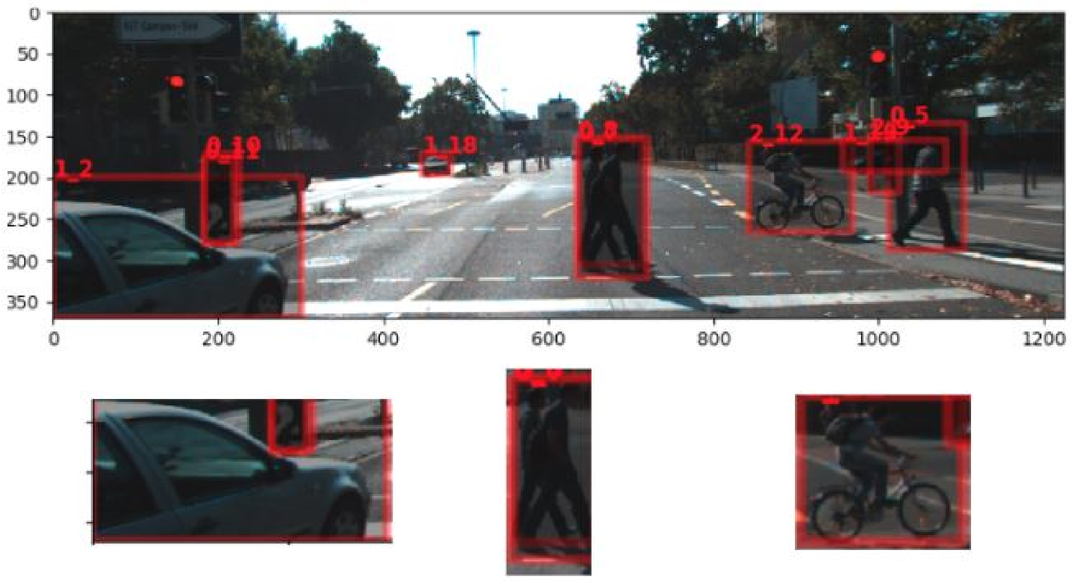


Figure 6.2: RGB image patches.

6.2.3 Scene

Including information about the scene and context can potentially help to understand the type of traffic scene where the prediction is happening. These images could be raw RGB images or pre-processed semantic maps of the scenes, as

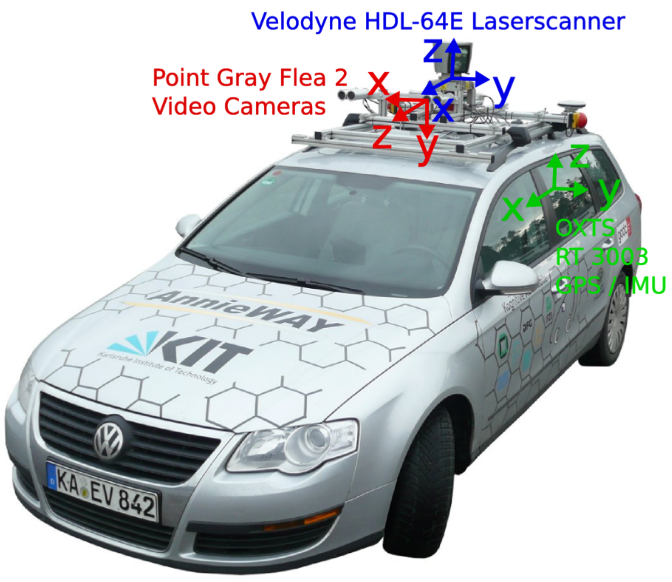


Figure 6.3: Ego-vehicle features [26]

presented in Figure 6.4.

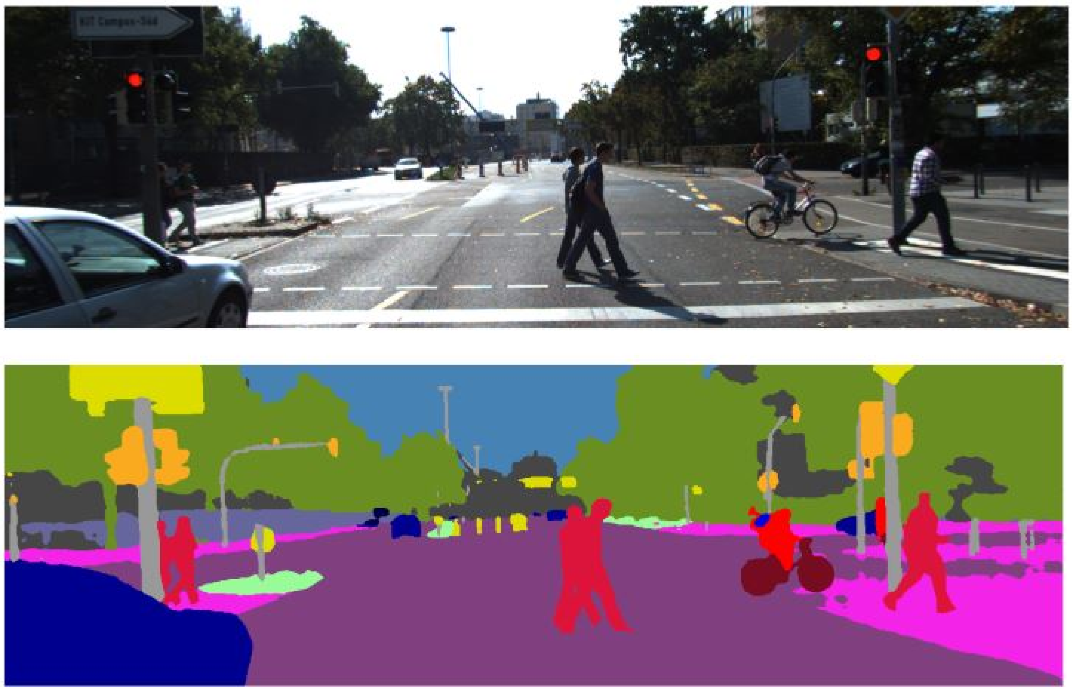


Figure 6.4: Scene features.

6.2.4 Interaction aware

Taking into account the position of other objects is an important feature for path prediction that is called interaction aware features. In this work three representations of the position of the objects in a scene are used – a grid map, a polar map (relative position), and the local map of objects.

Grid Map

As in [98], the grid map was implemented to encode the position of other objects around the object of interest in a rectangular grid, where each square of the grid represents one position. If no object is found in that position the square contains a value of 0. The Figure 6.5 shows an example of 6 frames (1 to 6) where each frame shows the RGB image of a scene, the created BEV representing that scene, and the grid map. In this example the object of interest is the cyclist (colored blue in the BEV), and the near object is the pedestrian (red). In the grid map visualisation, the position of the pedestrian object is shown in yellow.

Polar Map

The polar map was implemented as in [98]. In this type of map, instead of representing the position of the other objects in a rectangular grid, they are represented by polar coordinates, by the distance to the center of the object of interest and the angle, as depicted in the Figure 6.6. The polar map is divided into circles and by angle. The polar map shown in Figure 6.6 is divided in three circles and the total angle of the circumference (360° is divided into 4 sections. The positions of the objects are then represented by these circles and sections.

Local Bird's-Eye View Map

The Local BEV Map encodes the position of other objects in a scene, and assigns a color to each object type as depicted in Figure 6.7. This image map represents in pixels the position of the objects in the real world. The map encodes the type of each object in a specific color; red for pedestrian, green for vehicles, and blue for cyclists. The image in Figure 6.8 shows the global BEV map at different scales

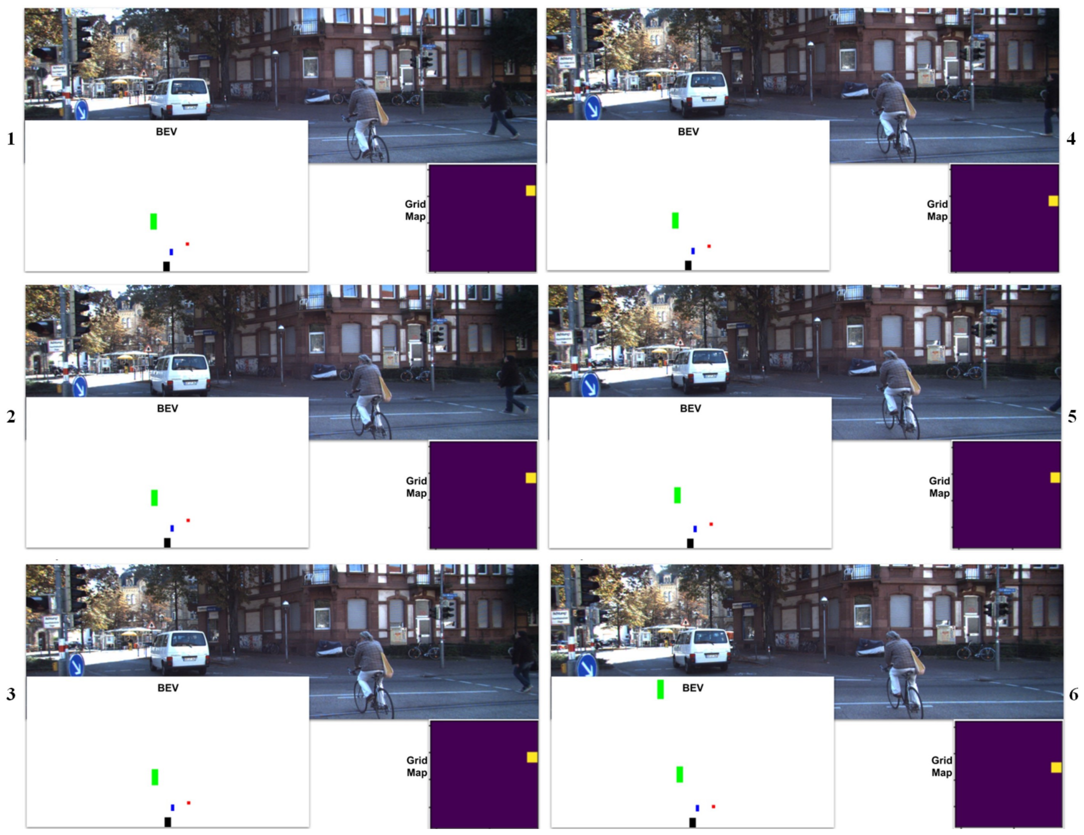
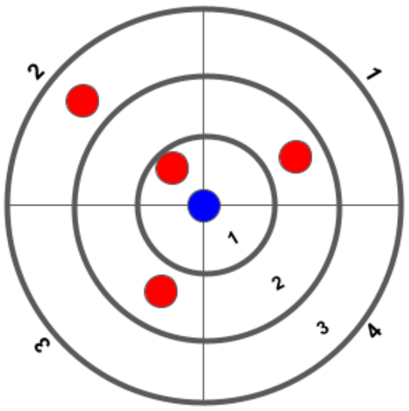
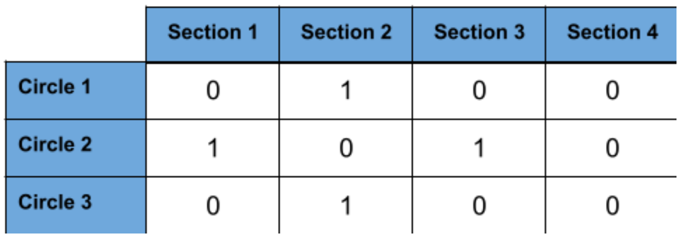


Figure 6.5: Grid map.





	Section 1	Section 2	Section 3	Section 4
Circle 1	0	1	0	0
Circle 2	1	0	1	0
Circle 3	0	1	0	0

Figure 6.6: Polar map.

to better represent the spatial position and separation of objects. In the example shown the two pedestrians cannot be distinguished at scale 1px:1m but start to become more distinct at scale 1px:0.5m.

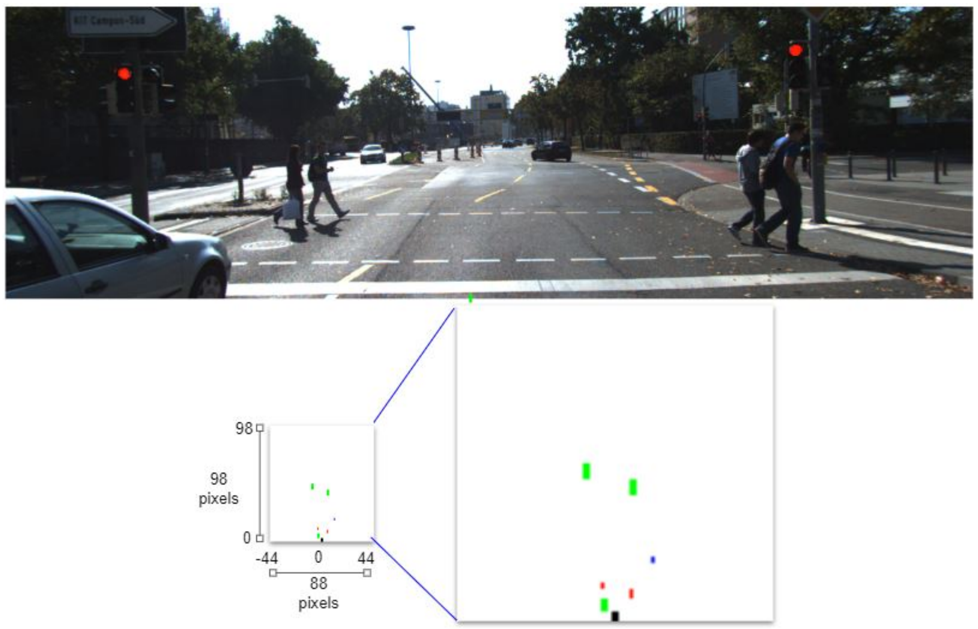


Figure 6.7: Interaction aware features: BEV map of objects in a scene. Red: pedestrian; Green: vehicles; Blue: cyclist.

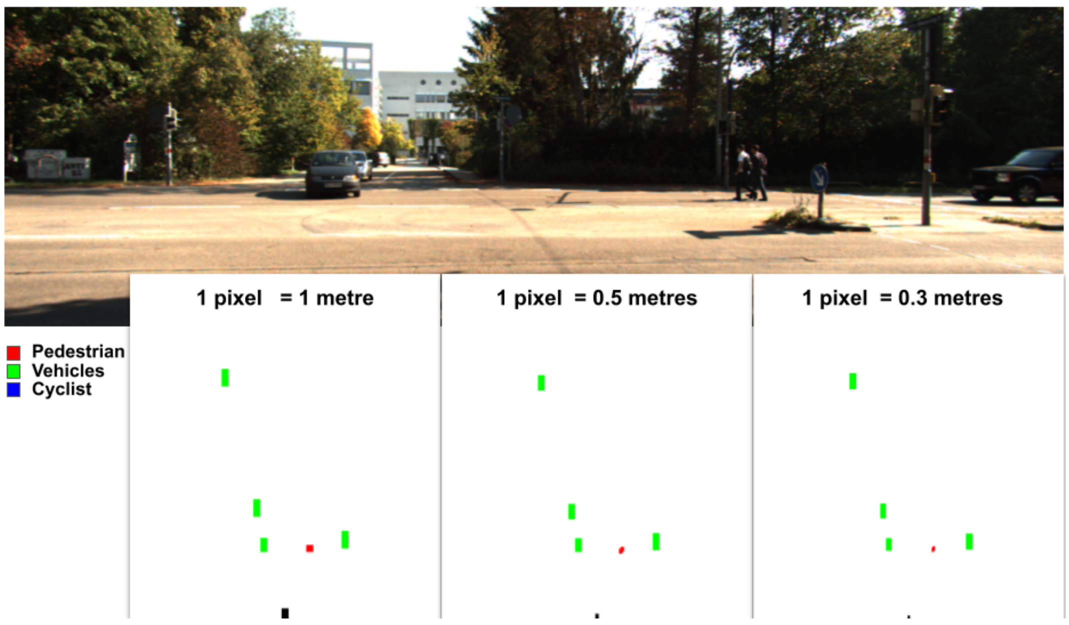


Figure 6.8: Global BEV Scales.

However, the BEV contains *all* the objects in a scene, but for predicting the path of an object, adjacent or near by objects are the most influential. Therefore, a local BEV is extracted from the global BEV. This is illustrated in Figure 6.9. It shows the lateral image of a scene, the global BEV representing that scene and the local BEV of the object of interest.

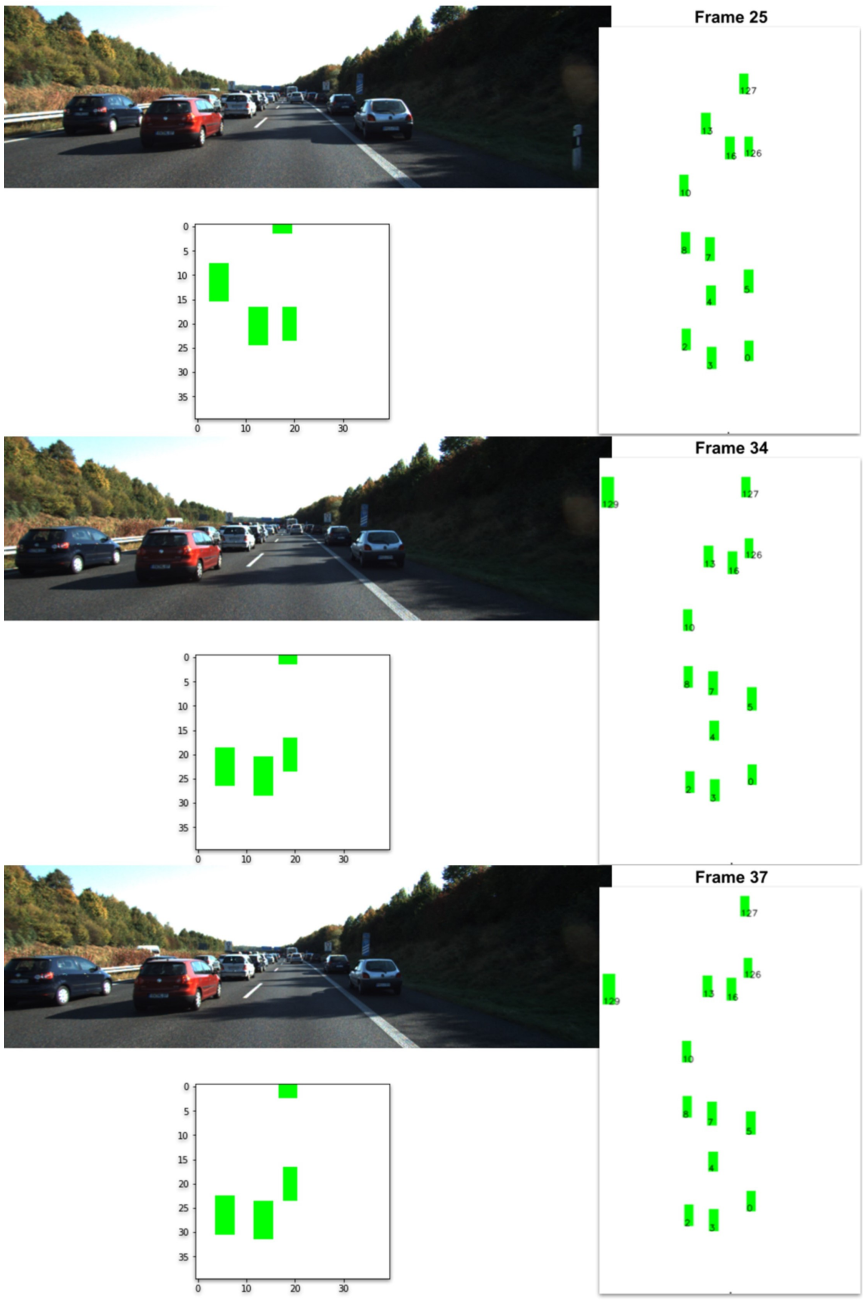


Figure 6.9: Local BEV for Object ID 126.

6.3 Data Definition

A path, P, is a set of tracks, tr, that contains information such as $tr(x, y)$ position (coordinates) of an object that travels a given space or scene, $P = \{tr_{t1}, tr_{t2}, ...tr_{tlength}\}$. However, besides the spatial location of the object, there are other features that can be extracted from the object itself, the ego vehicle, the other objects and the scene. This information can be also extracted sequentially so a track can be represented as a set of features such as $tr(x, y,$ $RGBObject$, $OtherObjects$, $EgoVehicle$, $Scene$, $Time)$.

In this work we extracted all this information and it was used to predict the future path of objects. Again, we used the sliding window approach to create observed tracklets: $tr^O = [tr^O_{t1}, tr^O_{t2}, ..., tr^O_{tobs}]$ and its respective ground truth tracklet: $tr^G = [tr^G_{t1}, tr^G_{t2}, ..., tr^G_{tpred}]$. The predicted vector of each tr^O is called $tr^P = [tr^P_{t1}, tr^P_{t2}, ..., tr^P_{tpred}]$. The aim of this chapter is to predict a tr^P based on the observed tracks tr^O, as illustrated in Figure 6.10.

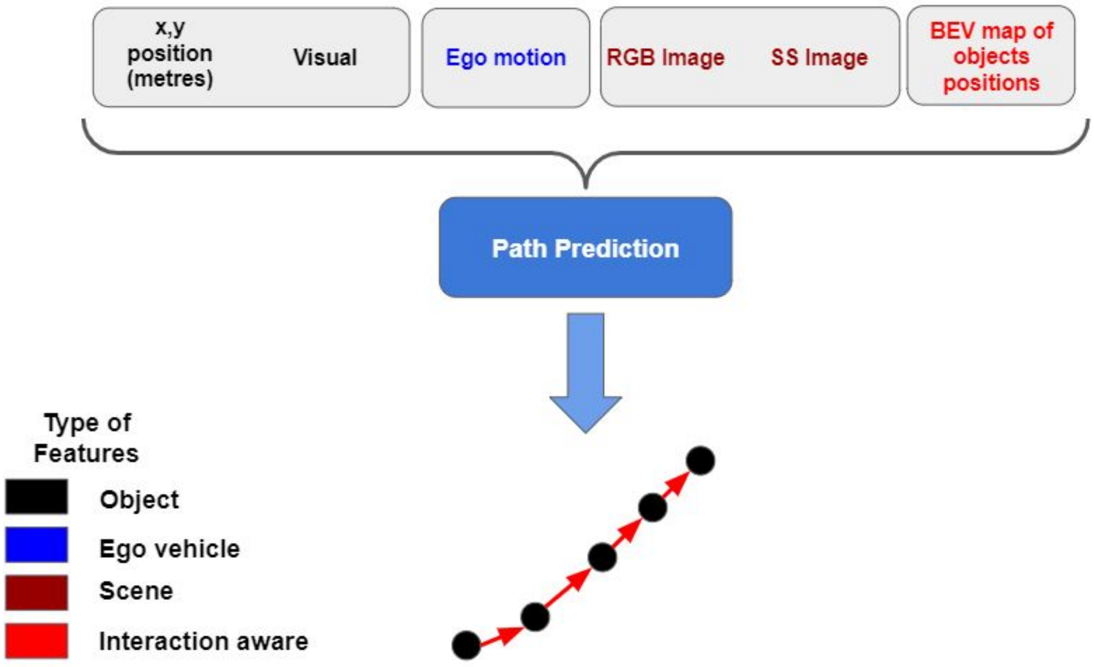


Figure 6.10: Path prediction using contextual information.

6.4 Approach

Now that we are dealing with more data, including numerical and visual information, and from different sources, we have to face three problems: a) Feature extraction, b) Multimodal data and c) Fusion strategies. This section presents a description of these problems and how they were addressed in this research. Firstly, explaining first how features were extracted from images using CNN models. Secondly, describing how models with multiple inputs can be created. And thirdly, a description of some fusion strategies that help to combine the different contextual features is given. Finally, the architectures of the models are given along with the steps followed during training and testing to obtain the future predicted path of objects.

6.4.1 Feature Extraction

Feature extraction has an important role in several applications in the computer vision area and CNN models have shown good performance on feature extraction for several tasks such as image classification, fine grained recognition and attribute detection [44] and several pre-trained models exist such as AlexNet, VGG16, VGG19, etc. In this application we are still dealing with sequential data so LSTM architectures are used. The combination of CNN and LSTMs are called CNN+LSTMs models. Besides CNN+LSTMs models there is a new variant called ConvLSTMs that fuse CNN+LSTMS internally. The Figure 6.11 and 6.12 depicts a general idea of the two approaches.

In this work, because of its wider use in the literature, the CNN+LSTMs approach was used. It is also important to mention that this application is different to classification problems. In this work the CNN models are trained from scratch together with an LSTM specifically for path prediction. So, it is expected that the CNN model used should capture features relevant for this task [98].

One of the limitations when dealing with a sequence of images is that the CNN models could be highly resource consuming, particularly with large images.

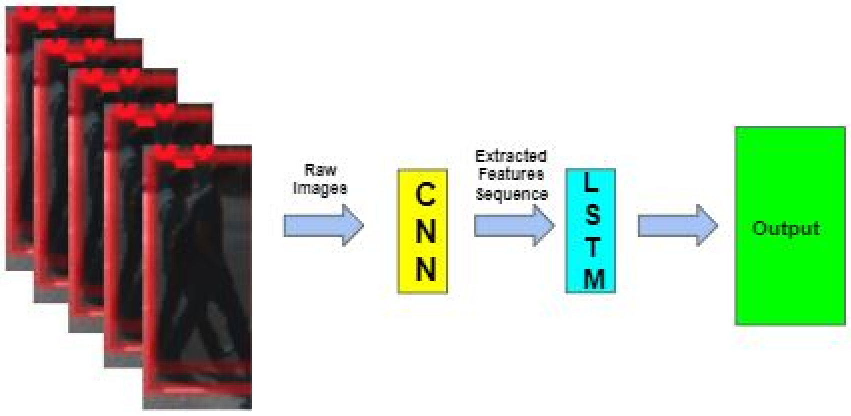


Figure 6.11: CNN+LSTM general approach

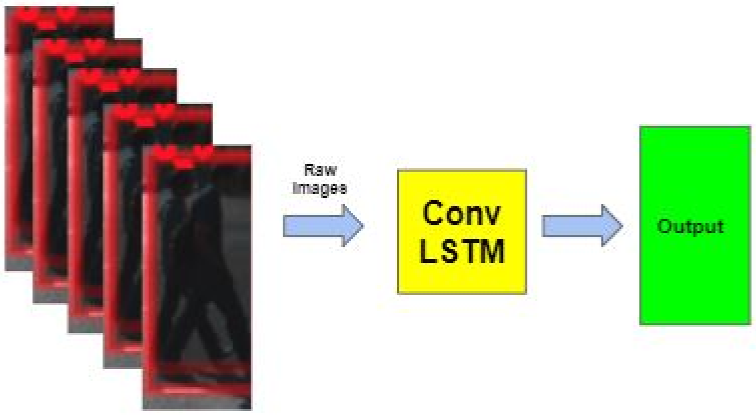


Figure 6.12: ConvLSTM general approach

Therefore in this work we focus on working with subsampled images of sizes such as 40×40, 64×64 and 124×37 pixels based on the size of the map. For that reason some research was done on tiny images classification. An interesting work is detailed in [112], where they present a 4Block-4CNN model that performs well on tiny images of 32x32 pixels from the CINIC-10 dataset which is a combination of CIFAR-10 and Tiny ImageNet. The model shown in this work can be seen as a deep model since it has 16 CNN layers grouped in 4 blocks. Another related work is done in [55] where they used the Tiny ImageNet dataset with images of 64x64. In this work they conclude that the use of deep models (8CNNs) lead to better performance. Similar conclusions were made in [100, 126]. Something in common in previous works is the use of several layers of CNNs in the model architecture.

Besides the use of deep CNN models, another aspect to consider is the use of Dropout or Batch Normalization. Based on the conclusions of the works presented in [48, 117], it was decided to use batch normalization.

Another point to consider is the use of a Global Average Pooling Layer (GAP) that allows the model to generate more generic features and also helps to expose the regions where the CNN are paying more attention [65]. This is a desirable characteristic since we are not aiming to classify an image but to predict the future path of objects based on them.

Taking into account the experience of previous works, three CNN models were created to process each type of image in this work. Table 6.1, Table 6.2 and Table 6.3 detail the models used to process the object image, the interaction-aware image and the scene image respectively.

For comparison purpose, based on information in [65] – “The responses from the higher-level layers of CNN (e.g.,fc6, fc7 from AlexNet) have been shown to be very effective generic features with state-of-the-art performance on a variety of image datasets” (p.5) – and for its use in [98], we decided to use AlexNet to compare performance with our proposed CNN models. AlexNet was also used with a final GAP layer to process the object image, the interaction-aware image map and the scene image as described in Table 6.4, Table 6.5 and Table 6.6.

Table 6.1: 4block3convGAP for object image.

Layer	Kernels	Kernel Size	Strides	Output shape
conv1	32	3x3	1	64x64x32
conv2	32	3x3	2	32x32x32
conv3	32	3x3	1	32x32x32
maxPool1	-	3x3	2	16x16x32
batchNorm1	-	-	-	16x16x32
conv4	64	3x3	1	16x16x64
conv5	64	3x3	1	16x16x64
conv6	64	3x3	2	8x8x64
maxPool2	-	3x3	2	4x4x64
batchNorm2	-	-	-	4x4x64
conv7	128	3x3	1	4x4x128
conv8	128	3x3	1	4x4x128
conv9	128	3x3	1	4x4x128
batchNorm3	-	-	-	4x4x128
conv10	256	3x3	1	4x4x256
conv11	256	3x3	1	4x4x256
conv12	256	3x3	1	4x4x256
GAP	-	-	-	256
dense1	-	-	-	256

Table 6.2: 4block3convGAP for interaction-aware image.

Layer	Kernels	Kernel Size	Strides	Output shape
conv1	32	3x3	1	40x40x32
conv2	32	3x3	2	20x20x32
conv3	32	3x3	1	20x20x32
maxPool1	-	3x3	2	10x10x32
batchNorm1	-	-	-	10x10x32
conv4	64	3x3	1	10x10x64
conv5	64	3x3	1	10x10x64
conv6	64	3x3	1	10x10x64
maxPool2	-	3x3	2	5x5x64
batchNorm2	-	-	-	5x5x64
conv7	128	3x3	1	5x5x128
conv8	128	3x3	1	5x5x128
conv9	128	3x3	1	5x5x128
batchNorm3	-	-	-	5x5x128
conv10	256	3x3	1	5x5x256
conv11	256	3x3	1	5x5x256
conv12	256	3x3	1	5x5x256
GAP	-	-	-	256
dense1	-	-	-	256

Table 6.3: 4block3convGAP for scene image.

Layer	**Kernels**	**Kernel Size**	**Strides**	**Output shape**
conv1	32	3x3	1	37x124x32
conv2	32	3x3	2	19x62x32
conv3	32	3x3	1	19x62x32
maxPool1	-	3x3	2	9x31x32
batchNorm1	-	-	-	9x31x32
conv4	64	3x3	1	9x31x64
conv5	64	3x3	1	9x31x64
conv6	64	3x3	2	5x16x64
maxPool2	-	3x3	2	2x8x64
batchNorm2	-	-	-	2x8x64
conv7	128	3x3	1	2x8x128
conv8	128	3x3	1	2x8x128
conv9	128	3x3	1	2x8x128
batchNorm3	-	-	-	2x8x128
conv10	256	3x3	1	2x8x256
conv11	256	3x3	1	2x8x256
conv12	256	3x3	1	2x8x256
GAP	-	-	-	256
dense1	-	-	-	256

Table 6.4: AlexNetGAP for object image.

Layer	**Kernels**	**Kernel Size**	**Strides**	**Output shape**
conv1	96	11x11	4	16x16x96
maxPool1	-	3x3	2	7x7x96
batchNorm1	-	-	-	7x7x96
conv2	256	5x5	1	7x7x256
maxPool2	-	3x3	2	3x3x256
batchNorm2	-	-	-	3x3x256
conv3	384	3x3	1	3x3x384
conv4	384	3x3	1	3x3x384
conv5	256	3x3	1	3x3x256
GAP	-	-	-	256
dense1	-	-	-	256

Table 6.5: AlexNetGAP for interaction-aware image.

Layer	**Kernels**	**Kernel Size**	**Strides**	**Output shape**
conv1	96	11x11	4	10x10x96
maxPool1	-	3x3	1	8x8x96
batchNorm1	-	-	-	8x8x96
conv2	256	5x5	1	8x8x256
maxPool2	-	3x3	2	3x3x256
batchNorm2	-	-	-	3x3x256
conv3	384	3x3	1	3x3x384
conv4	384	3x3	1	3x3x384
conv5	256	3x3	1	3x3x256
GAP	-	-	-	256
dense1	-	-	-	256

Table 6.6: AlexNetGAP for scene image.

Layer	**Kernels**	**Kernel Size**	**Strides**	**Output shape**
conv1	96	11x11	3	13x42x96
maxPool1	-	3x3	2	6x20x96
batchNorm1	-	-	-	6x20x96
conv2	256	5x5	1	6x20x256
maxPool2	-	3x3	2	2x9x256
batchNorm2	-	-	-	2x9x256
conv3	384	3x3	1	2x9x384
conv4	384	3x3	1	2x9x384
conv5	256	3x3	1	2x9x256
GAP	-	-	-	256
dense1	-	-	-	256

6.4.2 Multimodal Data

Multimodal data refers to the concept of having multiple types or modes of data. Our model must be able to ingest this "multimodal data" and make (accurate) predictions using it.

The information we are processing comes from different sources and in different types as shown in Figure 6.10. We want to build an end-to-end model capable of processing all this information. To achieve this requirement, we are using the Keras functional API [1] which is a way to create models that is more flexible than the Keras Sequential API [2]. The functional API can handle models with non-linear topology, models with shared layers, and models with multiple inputs or outputs. Figure 6.13 shows the architecture of a model with several inputs and outputs.

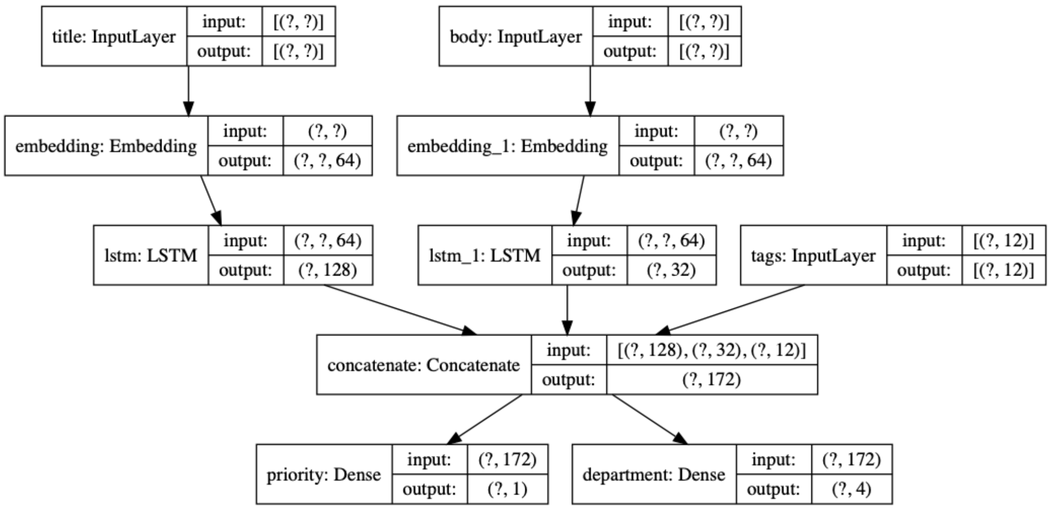

title: InputLayer | input: [(?, ?)] | output: [(?, ?)]
body: InputLayer | input: [(?, ?)] | output: [(?, ?)]
embedding: Embedding | input: (?, ?) | output: (?, ?, 64)
embedding_1: Embedding | input: (?, ?) | output: (?, ?, 64)
lstm: LSTM | input: (?, ?, 64) | output: (?, 128)
lstm_1: LSTM | input: (?, ?, 64) | output: (?, 32)
tags: InputLayer | input: [(?, 12)] | output: [(?, 12)]
concatenate: Concatenate | input: [(?, 128), (?, 32), (?, 12)] | output: (?, 172)
priority: Dense | input: (?, 172) | output: (?, 1)
department: Dense | input: (?, 172) | output: (?, 4)

Figure 6.13: Keras functional API: models with multiple inputs and outputs.

6.4.3 Fusion Strategies

The availability of features from different sources poses a challenge that is still open to further investigation. Fusion strategies define the way that different streams of available features will be joined in the model architecture. Three main fusion

[1] https://keras.io/guides/functional_api/
[2] https://keras.io/api/models/sequential/

strategies are described in [88, 111]:

Early fusion: the stream of features are combined at the beginning of the network, at the input level, before any layer such as CNN or LSTM [88].

Mid-level fusion: the information exchange takes place at the feature maps level of the intermediate network layers, so that useful early feature extraction/encoding are taken into account. First, separate streams of features are processed at early layers and then they are fused into a joint model in a later stage [111].

Late fusion: this strategy consists of separate streams of features, where no interaction or information exchange is carried out between them. The fusion is done only in the final prediction layer (i.e., after the softmax normalization) where the confidences for each class, or the predicted values, are averaged between the streams [111].

A clear illustration is given by [88] and shown in Figure 6.14 and more complex multimodal data fusion deep learning models are recently analysed in [116].

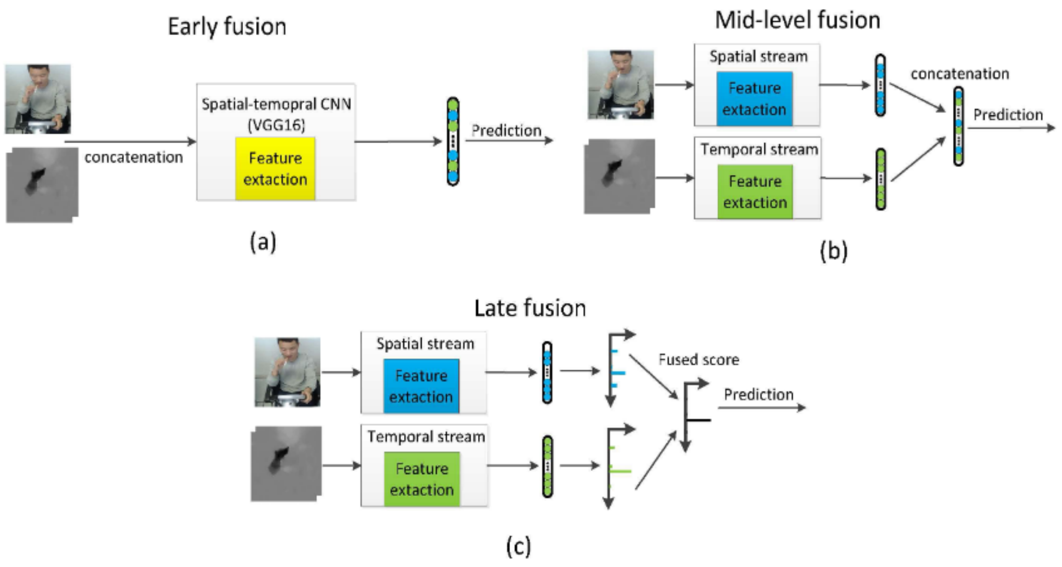


Figure 6.14: Illustration of different fusion strategies [88].

6.4.4 Model Architecture

The main focus of this research is to measure the performance gain of using more features to predict the object's path and two foundational architectures were used:

Vanilla LSTM: an LSTM using the default Keras' configuration.

Encoder-Decoder: the encoder is comprises a vanilla LSTM for numerical data and a CNN+LSTM for image data. The decoder is a vanilla LSTM that is fed with the features or concatenation of features from the encoder.

Each LSTM has 128 units using the default Keras' configuration. The Adam optimizer was used, also with the default values. For regression loss, Mean Squared Error (MSE) was used. The architecture of the models change according to the features and are illustrated in their respective sections below.

6.5 Experimental Setup

The dataset used to evaluate the models was KITTI. KITTI because of its realistic scenes, such as highways, inner city, vehicles standing, vehicle moving, its different objects and the labeled data. All this from cameras mouthed on a vehicle which is the main focus of this research work. Most of the datasets that provide data for forecasting give only the positions of the objects in world coordinates and provide a map of the scenes. However, these datasets do not provide RGB images of the scenarios and ego vehicle features as KITTI does. KITTI provides information from different type of sensors that can be fused and used together, which is desirable in this research work where we are focused in using different type of information present in a normal traffic scene. Besides positional information of the objects in metres, KITTI also provides different contextual information that surrounds a vehicle such as visual information of the objects, ego motion, different type of scenes and other objects position. Some of the new datasets provide more information of the surrounding of a vehicle. However, they started to provide this information too recently to be included in this research.

The size of the dataset is 20,141 samples with class distribution of cyclists (802), pedestrians (6,312) and vehicles (13,027). As in chapter 3, the models were evaluated on three objects – pedestrian, vehicles, and cyclists – for a prediction horizon (P.H.) of ± 20 steps. The available dataset is not big and no standard test/train split is available. For this reason a 5-fold cross validation was done and the mean results are reported. All experiments were run for 1000 epochs, except for those where an initial evaluation of CNN models was done (300 epochs). The batch size used was 32 due to hardware limitations and because using small batch size leads to a better trained model.

The same sliding window method as reported in previous chapters was used to create the observed tracklets, $tr^O = [tr^O_{t1}, tr^O_{t2}, ..., tr^O_{tobs}]$ and its respective ground truth tracklet, $tr^G = [tr^G_{t1}, tr^G_{t2}, ..., tr^G_{tpred}]$.

6.5.1 Data Pre-processing

Each type of data was processed as follows:

Object: (x,y,z) position in metres were extracted. RGB features: patches of the objects were extracted and resized to 64×64 pixels.

Ego vehicle: orientation, velocity in [x,y,z], acceleration in [x,y,z] features were extracted from the oxt files which are the dynamics of the ego vehicle.

Scene: RGB images from the scenes were resized to 124×37 pixels.

Interaction aware: for grid and polar map they were flattened first to feed an LSTM. The local image BEV map was resized to 40×40 pixels.

The mentioned information is provided by KITTI in different files, to able of using these features together, all the information was synchronized taking as time stamp/ID reference the frame number of each measurement.

6.5.2 Training

During training, a cross validation with $K = 5$ was done and for each fold a training and test split was done with 70% of the data used for training and the rest for testing. Also, to feed the model correctly, the next steps were performed:

1. Specify the length of tracklets to be processed.
2. Select the features to use. Besides x, y positional information, other features such as object image, ego-vehicle information, scene image and interaction-aware information can be used.
3. Scale data [0,1]: un-scaled input variables can result in a slow or unstable learning process, whereas un-scaled target variables on regression problems can result in exploding gradients causing the learning process to fail. Images were divided by 255 before being fed to the CNN models.
4. Split data into observed tracklet, tr^O, and ground truth (tracklet) path to be predicted, tr^G.
5. tr^O is shaped as [$Nsamples$, *tobs*, $features$-*in*-tr^O] and tr^G as [$Nsamples$, *OutputLength*]. Where $Nsamples$ is the number of samples in the dataset, *tobs* is the number of tracks in each observed tracklet (tr^O), $features$-*in*-tr^O is the number of features in each observed track. Finally, *OutputLength* is the size of the output of the model. In this case, *OutputLength* $=$ *tpred* $*$ $features$-*in*-tr^G. Here *tpred* is the steps to predict in the future (the prediction horizon) and $features$-*in*-tr^G is the number of features to predict in each step.

6.5.3 Single Path Prediction

In chapter 5, it was shown that LSTM architectures can be extended to predict multiple paths by combining them with Mixture Density Models (MDMs) as a final layer. However, in this chapter a single path was predicted to better analyse the impact of the contextual features. To obtain a single path, the output of the model was processed as follow:

1. The model outputs a single array, tr^P, per tr^O of size $OutputLength$.
2. tr^P is re-shaped to $[tpred, features\text{-}in\text{-}tr^G]$.

6.6 Exploration of Ego-vehicle Features

Ego-vehicle features are an interesting source of information that can help to improve the path prediction results. Besides, KITTI provide this features with out any pre-processing stage that can be used to enrich the tracklets. So in this first set of experiments the dynamics of the vehicle (ego vehicle) is fused with the position of the objects. The KITTI dataset provides several dynamics and from them the following mentioned below were selected; They were selected because are more related to the movement of the vehicle on the ground:

Yaw: heading (rad), 0 = east, positive = counter clockwise (-pi..pi)

VF: forward velocity, i.e. parallel to earth-surface (m/s)

VL: leftward velocity, i.e. parallel to earth-surface (m/s)

VU: upward velocity, i.e. perpendicular to earth-surface (m/s)

AF: forward acceleration (m/s^2)

AL: leftward acceleration (m/s^2)

AU: upward acceleration (m/s^2)

The experiment consists in the use of different combinations of the ego-vehicle features and assessment the performance of the model when using each set of combinations. For these experiments the model used was the vanilla LSTM. The results are presented for each object independently and one where all objects are included (multiclass). The ADE and FDE metrics used are defined in equation 2.11 and equation 2.12.

An initial exploration of using more features along with x, y positional information, which is related to **RQ3** was done in chapter 5. In this chapter 6, these experiments provide a second exploration to ***Q3: Are Long Short-Term Memory (LSTM) architectures suitable for sequential and enriched trajectory information?*** by adding ego-motion information to the vector of features to process.

6.6.1 Results

The results are shown in Figure 6.15. A cross validation with $K = 5$ was done and the mean predicted path error values are depicted (in metres). It can be seen that the ego-vehicle features do not improve the performance of the model for short P.H. (row 1 and 2) but they do for longer P.H. (row 3 and 4). This is clear for the object vehicle and also for the multiclass prediction though this can be attributed to the higher number of vehicle objects compared with cyclists and pedestrians.

It can also be noted that the combination of $[X, Y, VF, FL]$ and $[X, Y, VF, FL, AF, AL]$ leads to better overall performance. The reason that these features improve the performance mostly for longer P.H. could be because the error is higher when we predict more than 1s in the future, so more information is need to reduce that error. Also, the reason that forward velocity (VF), leftward velocity (VL) and its related acceleration (AF, AL) contribute more to reduce the error could be because those features are more related to the plane (birds-eye view map) where the prediction is taking place.

6.6.2 Discussion of Exploring Ego-vehicle Features

This exploration of ego-vehicle information as extra features combined with solely positional information, gave us an initial sense of how the combination of features may help to reduce the errors in the prediction task. The following experiments

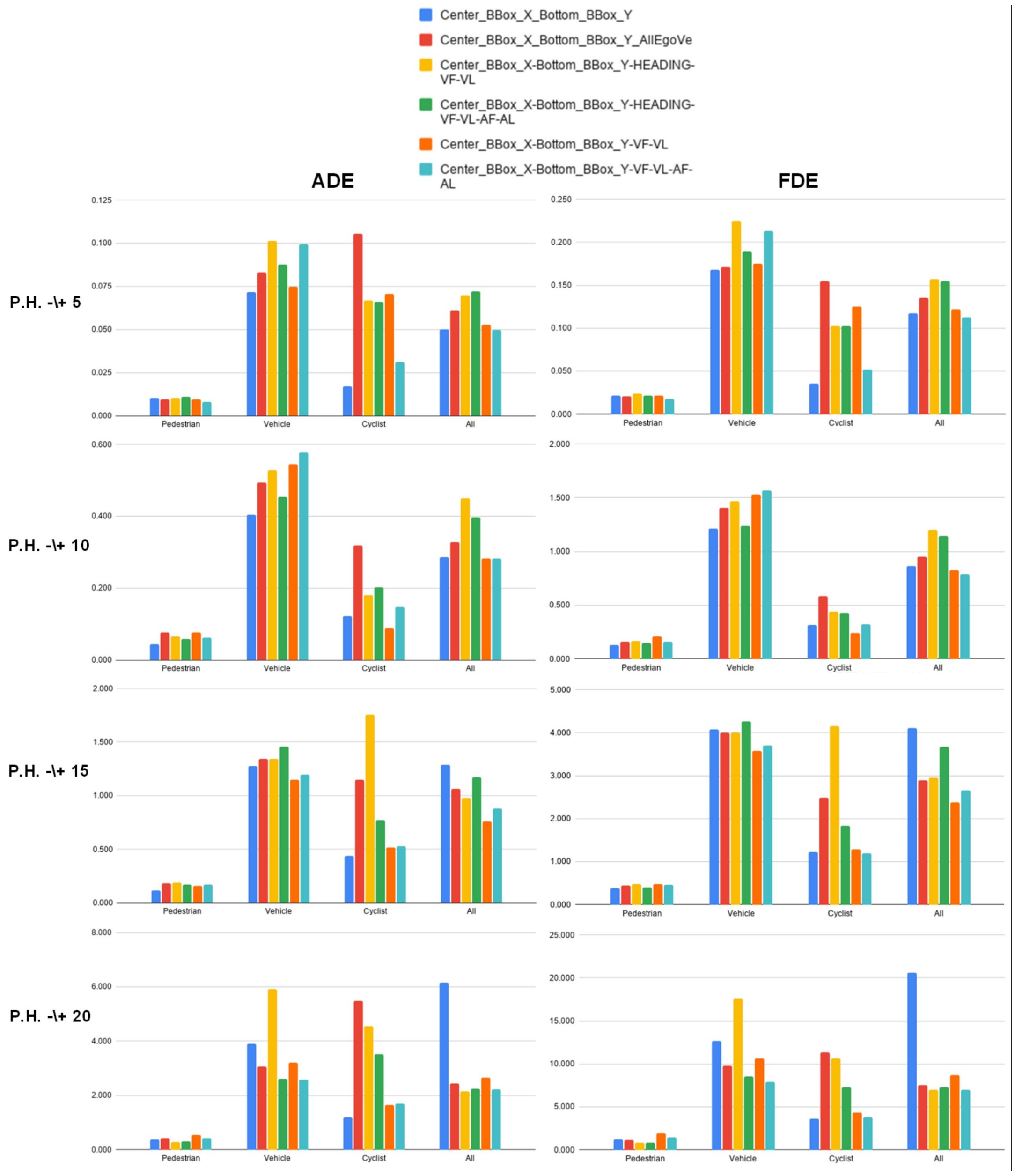


Figure 6.15: X, Y and ego-vehicle features early fusion.

expand this to use visual features from images, which does increase the time to train a model and, based on the learnings here, some decisions were made to be applied to the next experiments:

- Perform experiments over the longer prediction horizon, which is ± 20.
- Use the dataset that includes all types of objects.

- Use the metrics from the equation 2.13 and equation 2.14 which use Euclidean Distance.
- These experiments used a basic fusion strategy (early fusion), since we only concatenated the x, y features with the ego-vehicles information at the input level. This opens a new question about the use of other fusion strategies and internal feature encoding/representation.

6.7 Exploration of Multimodal Features

Q3: Are Long Short-Term Memory (LSTM) architectures suitable for sequential and enriched trajectory information? is deeply explored in this set of experiments where more features are combined with the positional information of the objects. Along with this, another question to answer in this research work is: **Q4** ***hlHow can contextual information of a scene be used to improve path prediction results?*** For that reason, the aim of this set of experiments is to find evidence that using more features with appropriated fusion strategies can lead to better performance in the prediction task. Also, specifically section **X,Y Features** relates to explore further **Q1** ***How should the observed object position (tracklets) be best represented?***, those experiments explore the representation of positional information in a latent space to feed an LSTM instead of using those features directly as raw features.

First, several combinations of pairs of features were tested, then, from here those features that led to better performance were selected to create a combination of three features:

1. x, y position.
2. x, y and ego-vehicle features.
3. x, y and object image.
4. x, y and interaction aware map.

5. x, y and scene image.

6. x, y and object image and ego-vehicle features.

In the previous set of experiments, **Exploration of ego-vehicle features**, a basic fusion of features was done. The fusion there consisted of only a concatenation of the two types of features at input level (early fusion), in the following experiments the following fusion strategies were evaluated:

1. Early fusion of raw features (RF).

2. Early fusion of latent space (LS) features.

3. Middle fusion of latent space (LS) features.

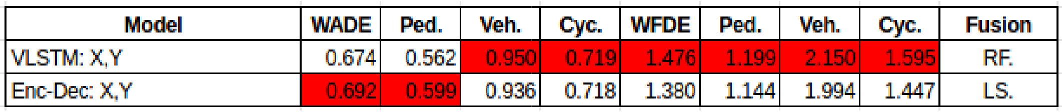
As mentioned in previous section, **Exploration of ego-vehicle feature**, which showed that using ego-vehicle features leads to better performance for longer P.H. and considering the time required to process images, the following experiments are done for the P.H. of ± 20 and on a dataset where all the objects are included. The metrics used are from the equation 2.13 and equation 2.14, for ADE and FDE respectively. The Euclidean variant was chosen as recent works such as [109] and some evaluation challenges, use Euclidean distance for both FDE and ADE.

The results are visualised using colour scaling where a red background color means that model got the worst performance and a white background color means that model got the best performance. This encoding of color makes easy to identify which model performs better for each metric and for each type of object. The first column of the figures contains the name of the model. The second column shows the weighted sum of ADE (WSADE), followed by the individual ADE for each type of object. The fifth column presents the weighted sum of FDE (WSFDE), followed by the individual FDE for each type of object.

6.7.1 Results

6.7.1.1 X,Y Features

This experiment consists of feeding an LSTM with raw features versus first encoding the raw features in a latent space by an LSTM (the encoder) and then feeding this to another LSTM (the decoder). The results are depicted in Figure 6.16. It can be seen that encoding the raw features in latent space improves performance over using the raw features to feed the Vanilla LSTM. This can be seen as two stacked LSTMs, however, the first LSTM is acting as an encoder of the raw features in its latent space and the second LSTM is leveraging this representation.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
VLSTM: X,Y	0.674	0.562	0.950	0.719	1.476	1.199	2.150	1.595	RF.
Enc-Dec: X,Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.

Figure 6.16: Raw features vs latent space features for x, y features.

The Vanilla LSTM and the Encoder-Decoder models used in these experiment are depicted in Figure 6.17 and Figure 6.18.

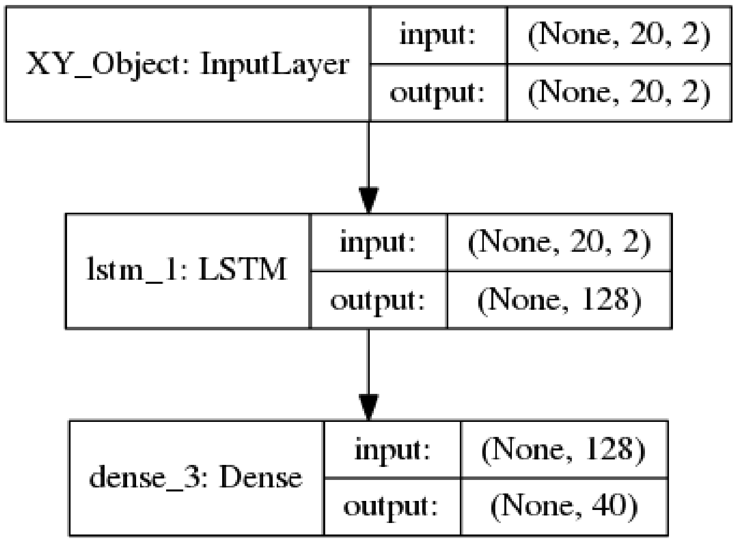


Figure 6.17: Vanilla LSTM for x, y features.

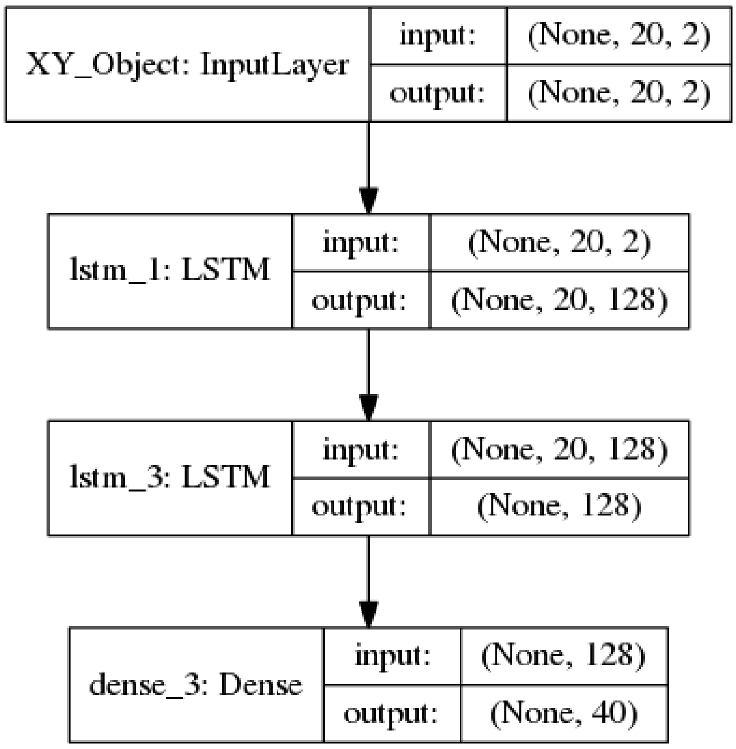


Figure 6.18: Encoder-Decoder for x, y features.

6.7.1.2 X,Y, Ego-vehicle Features

A first exploration of ego-vehicle features was done in section Exploration of ego-vehicle features. In these next experiments, a deeper exploration is carried out. Here, three combinations of ego-vehicle information were evaluated – 1. $[x, y, VF, VL]$, 2. $[x, y, VF, VL, AF, AL]$ and 3. $[x, y, HEADING, VF, VL, AF, AL]$ – since these three combinations gave better performance in the previous section. Also, to explore further these features, the three fusion strategies mentioned in the beginning of this section were tested. The results are shown in Figure 6.19. From there, two observations can be drawn:

1. Combination of features: the results indicate that using x, y and VF, VL, AF and AL reduce the error in the prediction.

2. Fusion strategy: in most of the cases using middle fusion of features' latent space improves performance. On the contrary, using the raw features directly

to feed the LSTM leads to a higher error.

Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y, VF, VL	0.766	0.545	0.987	1.148	1.694	1.152	2.193	2.671	Middle. LS
Enc-Dec: X, Y, VF, VL, AF, AL	0.673	0.519	0.909	0.866	1.442	1.077	2.019	1.878	Middle. LS
Enc-Dec: X, Y, HEADING, VF, VL, AF, AL	0.729	0.575	1.025	0.864	1.532	1.169	2.261	1.825	Middle. LS
VLSTM: X, Y, VF, VL	0.810	0.616	1.060	1.094	1.603	1.214	2.274	2.018	Early. RF.
VLSTM: X, Y, VF, VL, AF, AL	0.850	0.547	1.057	1.460	1.615	1.076	2.216	2.490	Early. RF.
VLSTM: X, Y, HEADING, VF, VL, AF, AL	0.770	0.581	1.016	1.044	1.585	1.103	2.122	2.367	Early. RF.
Enc-Dec: X, Y, HEADING, VF, VL, AF, AL	0.812	0.635	0.984	1.120	1.563	1.191	2.084	2.070	Early. LS
Enc-Dec: X, Y, VF, VL, AF, AL	0.748	0.556	0.941	1.078	1.526	1.089	2.015	2.234	Early. LS
Enc-Dec: X, Y, VF, VL	0.744	0.571	0.977	0.987	1.508	1.131	2.083	1.981	Early. LS

Figure 6.19: Early fusion raw features vs early fusion latent space vs middle fusion latent space for x, x and ego-vehicle features.

The models used to fuse x, y positional information and the ego-vehicle features in three different strategies – Early fusion no latent space, Early fusion latent space and, Middle fusion latent space – are presented in Figure 6.20, Figure 6.21, Figure 6.22, respectively. The figures show the fusion of four features – VF, VL, AF, AL of the ego vehicle – the fusion with the other two combinations – VF, VL and Heading, VF, VL, AF, AL – following the same architecture, the only difference is the number of features in each combination.

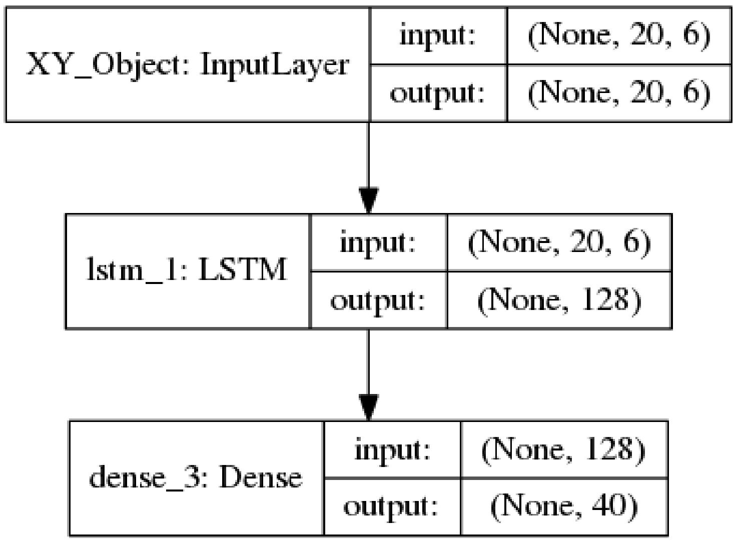


Figure 6.20: Early fusion no latent space for x, y and VF, VL, AF, AL features.

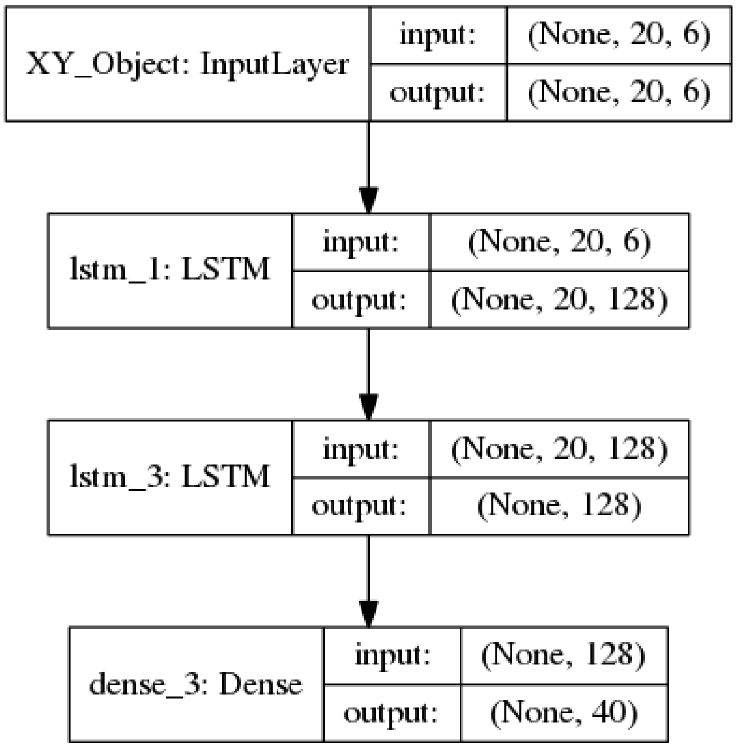


Figure 6.21: Early fusion latent space for x, y and VF, VL, AF, AL features.

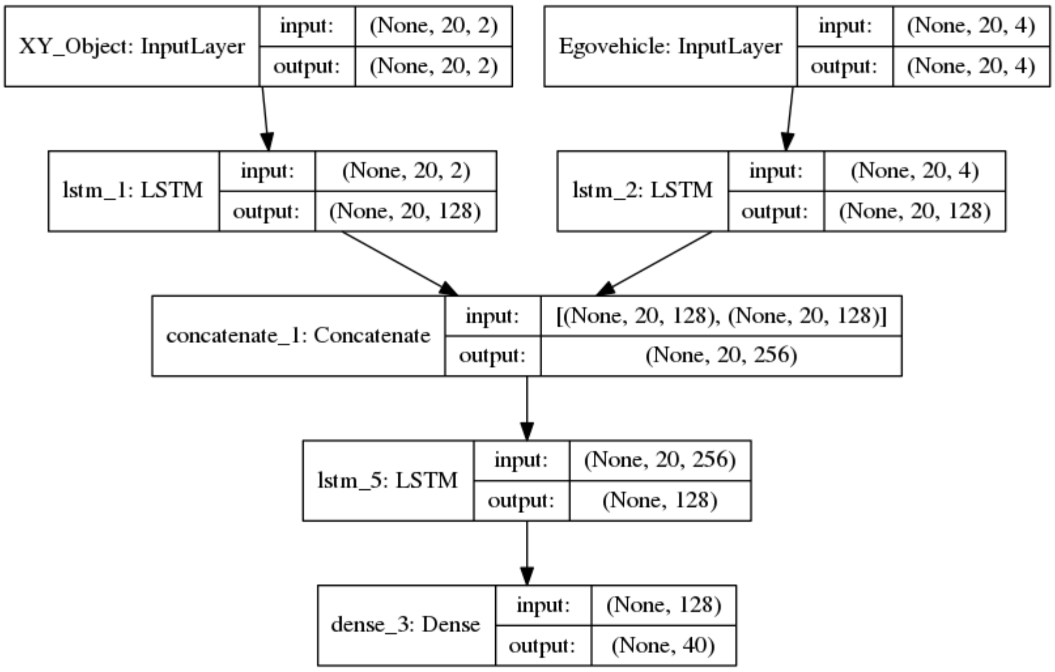
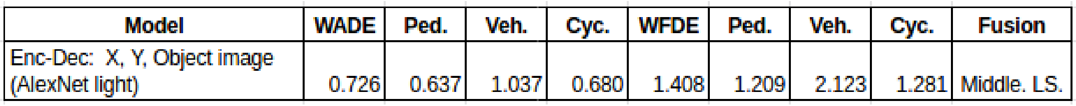


Figure 6.22: Middle fusion latent space for x, y and VF, VL, AF, AL features.

6.7.1.3 X,Y, Object Image

A well known methodology to extract features from images is the use of CNN models. These models act as a feature extractor then these features are fed to a classifier or neural network. The intial approach to include deep features was to use the model from SSLSTM [98], where they provide the code at [127]. Looking at the CNN they use, it is AlexNet without the two layers of Conv2D(384), which creates a shallow and narrow CNN model (AlexNet light). The results of this evaluation is shown in Figure 6.23. Comparing against the baseline (using only x, y positional information), there is no improvement, except for the object Cyclist where there is a small reduction in error.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y, Object image (AlexNet light)	0.726	0.637	1.037	0.680	1.408	1.209	2.123	1.281	Middle. LS.

Figure 6.23: Middle fusion latent space for x, y and Object Image using CNN from SSLSTM [98].

The next step was to find out if using a different CNN would improve the performance so based on available literature [112, 98, 65], we selected two CNNs as explained in subsection 6.4.1. The two models are:

4block3convGAP: this is a four block model, each block with 3 conv layers and a final GAP layer. It can be seen as a deep model.

AlexNetGAP: AlexNet with all layers and a final GAP layer. It can be seen as a shallow but wide model.

Due to the computational requirements, to initially test the performance of the models they were run for 300 epochs, using the k-folds numbers 1, 3 and 5. The results are shown in Figure 6.24. The results indicate that using the CNN model 4block3convGAP gives better results.

For the second stage of this experiment, the model 4block3convGAP was selected to be run for 1000 epochs and for all the k-folds. The results are shown in

Model / 300 Epochs	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y, Object image (4block3convGAP)	0.764	0.611	1.158	0.807	1.464	1.077	2.374	1.658	Middle. LS.
Enc-Dec: X, Y, Object Image (AlexNetGAP)	0.813	0.672	1.204	0.832	1.618	1.311	2.530	1.599	Middle. LS.

Figure 6.24: 4block3convGAP vs AlexNetGAP for object image.

Figure 6.25. It can be observed that the 4block3convGAP CNN improves significantly the performance of the model compared to using the CNN from SSLSTM. Additionally, the 4block3convGAP CNN also improves the performance against the base line model (using only x, y positional information). The improvement is mostly for FDE, which means an error reduction in the prediction of the final position of an object. The improvement reached by using the 4block3convGAP against the model from SSLSTM gives evidence that refining the CNN model used in the feature extraction leads to an improvement in the prediction task so future work can be done on this.

Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
Enc-Dec: X, Y, Object image (AlexNet light)	0.726	0.637	1.037	0.680	1.408	1.209	2.123	1.281	Middle. LS.
Enc-Dec: X, Y, Object image (4block3convGAP)	0.688	0.576	0.971	0.723	1.308	1.068	1.993	1.317	Middle. LS.

Figure 6.25: x, y and object image. x, y only vs AlexNet light vs 4block3convGAP for object image.

The Encoder-Decoder model used to fuse x, y positional information and object image is presented in Figure 6.26.

6.7.1.4 X,Y, Interaction-aware Map

Another important aspect to take into account when predicting the future path of an object is the position of other nearby objects. In this set of experiments, three types of maps were explored to include other objects positional information using a grid, a polar map, and one local BEV image map as explained in section 6.2.4.

The first two maps, grid and polar, can be seen as handcrafted features, since

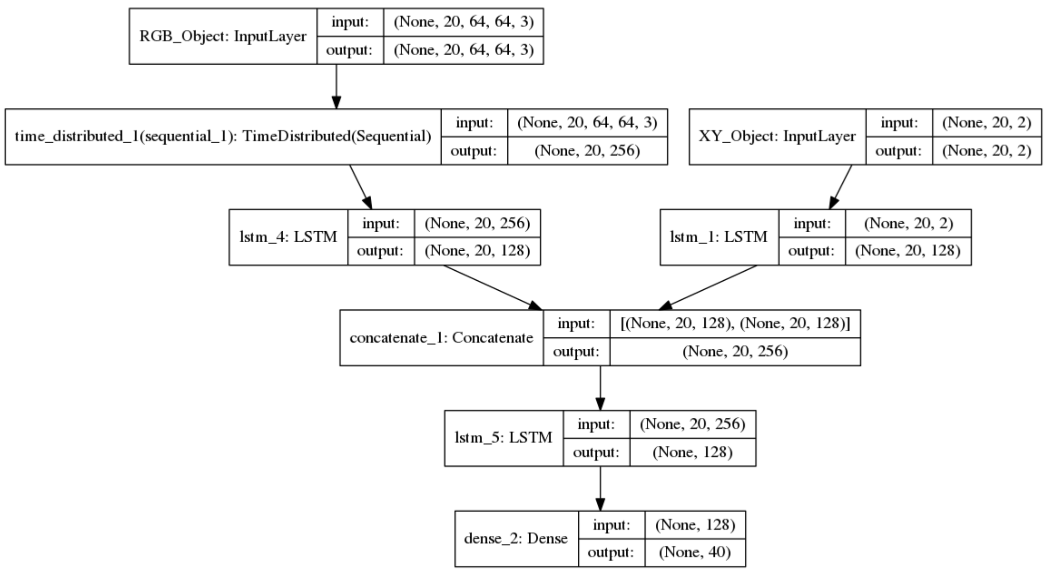
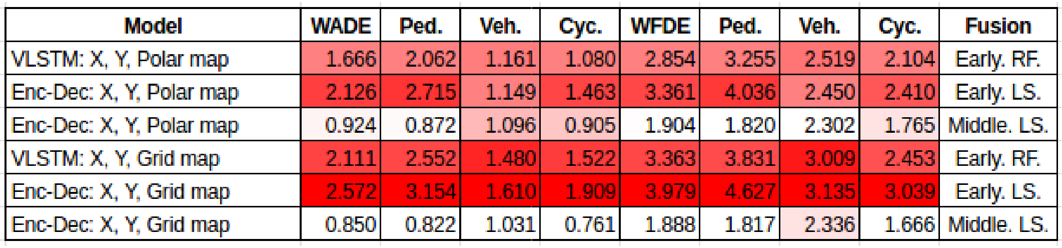


Figure 6.26: Fusion model for x, y and object image.

they were created and the maps are fed into the model with any previously extracted features. The results are presented in Figure 6.27. The following observations can be made:

1. Grid Map vs Polar Map: using a grid map shows better performance.
2. Fusion strategy: there is a significant error reduction when using middle fusion of features in the latent space. Early fusion however leads to a higher error. This is observed for both grid and polar map.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
VLSTM: X, Y, Polar map	1.666	2.062	1.161	1.080	2.854	3.255	2.519	2.104	Early. RF.
Enc-Dec: X, Y, Polar map	2.126	2.715	1.149	1.463	3.361	4.036	2.450	2.410	Early. LS.
Enc-Dec: X, Y, Polar map	0.924	0.872	1.096	0.905	1.904	1.820	2.302	1.765	Middle. LS.
VLSTM: X, Y, Grid map	2.111	2.552	1.480	1.522	3.363	3.831	3.009	2.453	Early. RF.
Enc-Dec: X, Y, Grid map	2.572	3.154	1.610	1.909	3.979	4.627	3.135	3.039	Early. LS.
Enc-Dec: X, Y, Grid map	0.850	0.822	1.031	0.761	1.888	1.817	2.336	1.666	Middle. LS.

Figure 6.27: Early fusion raw features vs early fusion latent space vs middle fusion latent space for x, y and interaction-aware features.

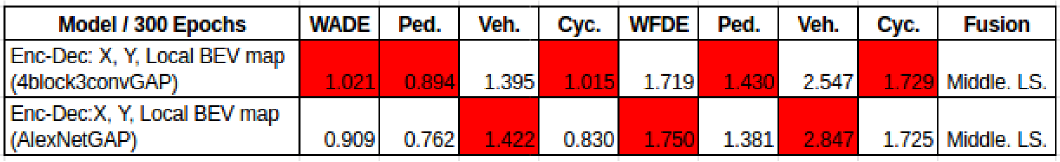
The third map, BEV, is an image which encode the position of other object, so to include the Local BEV in the prediction task, first this map was processed for a CNN model to extract deep features, then this deep features were fed to the model.

As in *X, Y, Object Image*, first, to find out if using a different CNN would improve the performance of the prediction, two CNNs models were selected as explained in subsection 6.4.1.

4block3convGAP: this is a four block model, each block with 3 conv layers and a final GAP layer. It can be seen as a deep model.

AlexNetGAP: AlexNet with all layers and a final GAP layer. It can be seen as a shallow but wide model.

The two CNNs were run for 300 epochs, using the k-folds numbers 1, 3 and 5 and the mean are reported. The results are shown in Figure 6.28. It can be seen that AlexNetGAP is slightly better than 4block3convGAP.

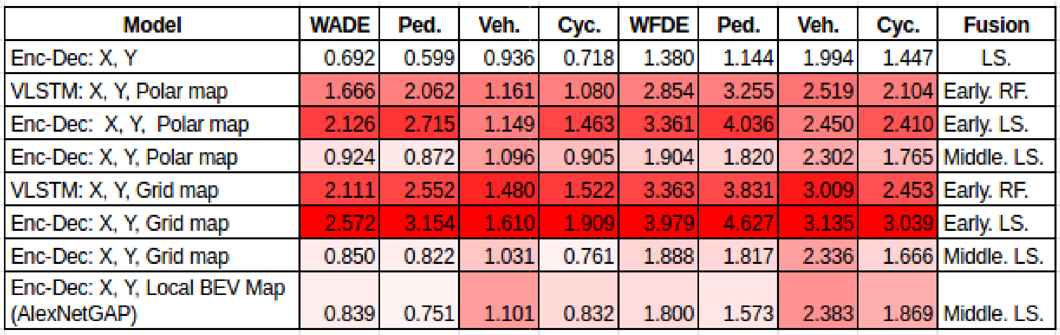


Model / 300 Epochs	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y, Local BEV map (4block3convGAP)	1.021	0.894	1.395	1.015	1.719	1.430	2.547	1.729	Middle. LS.
Enc-Dec:X, Y, Local BEV map (AlexNetGAP)	0.909	0.762	1.422	0.830	1.750	1.381	2.847	1.725	Middle. LS.

Figure 6.28: 4block3convGAP vs AlexNetGAP for Interaction-aware local BEV map.

The next step was to train the AlexNetGAP for 1000 epochs on the 5 folds. The results are depicted in Figure 6.29. It can be observed that using the local BEV with AlexNetGap improves the performance slightly over the use of grid with middle fusion on latent space. However, the use of local BEV with AlexNetGap, does not lead to error reduction over the base line model (using only x, y positional information). The reason that interaction-aware information did not add any improvement when combined with x, y positional information could be because KITTI dataset does not have many crowded scenes where the objects interact with each other.

Three different models were used to fuse positional information and the created handcrafted maps – early fusion with no latent space, early fusion with latent space and middle fusion with latent space. These are illustrated in Figure 6.30, Figure 6.31,



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
VLSTM: X, Y, Polar map	1.666	2.062	1.161	1.080	2.854	3.255	2.519	2.104	Early. RF.
Enc-Dec: X, Y, Polar map	2.126	2.715	1.149	1.463	3.361	4.036	2.450	2.410	Early. LS.
Enc-Dec: X, Y, Polar map	0.924	0.872	1.096	0.905	1.904	1.820	2.302	1.765	Middle. LS.
VLSTM: X, Y, Grid map	2.111	2.552	1.480	1.522	3.363	3.831	3.009	2.453	Early. RF.
Enc-Dec: X, Y, Grid map	2.572	3.154	1.610	1.909	3.979	4.627	3.135	3.039	Early. LS.
Enc-Dec: X, Y, Grid map	0.850	0.822	1.031	0.761	1.888	1.817	2.336	1.666	Middle. LS.
Enc-Dec: X, Y, Local BEV Map (AlexNetGAP)	0.839	0.751	1.101	0.832	1.800	1.573	2.383	1.869	Middle. LS.

Figure 6.29: x, y and Interaction-aware features. x, y only vs POLAR vs GRID vs AlexNetGAP for local BEV map.

Figure 6.32. The figures show the fusion with grid maps and the fusion with polar maps follows the same architecture. The only difference is number of features that compose each type of map.

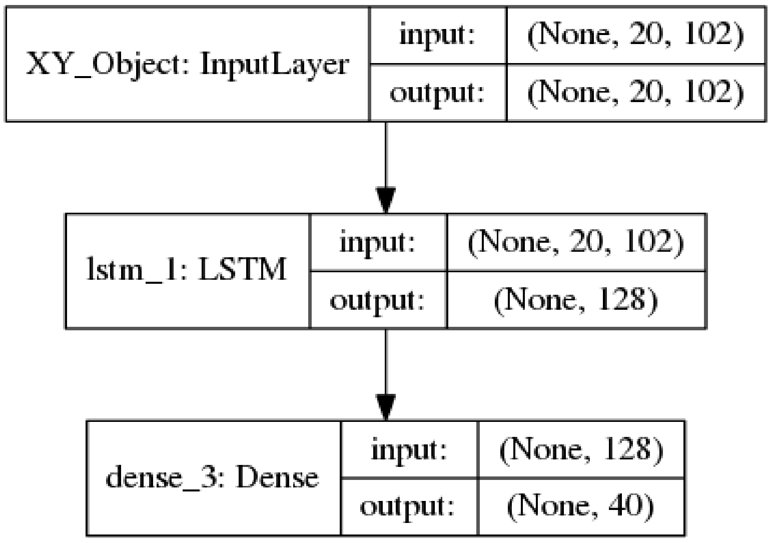


Figure 6.30: Early fusion no latent space for x, y and grid map.

The Encoder-Decoder model used to fuse positional information and the local BEV map image is presented in Figure 6.33.

6.7.1.5 X,Y, Scene Image

With the objective of including more context about traffic scenes where the path prediction task is happening, this experiment combines RGB images of the scenes

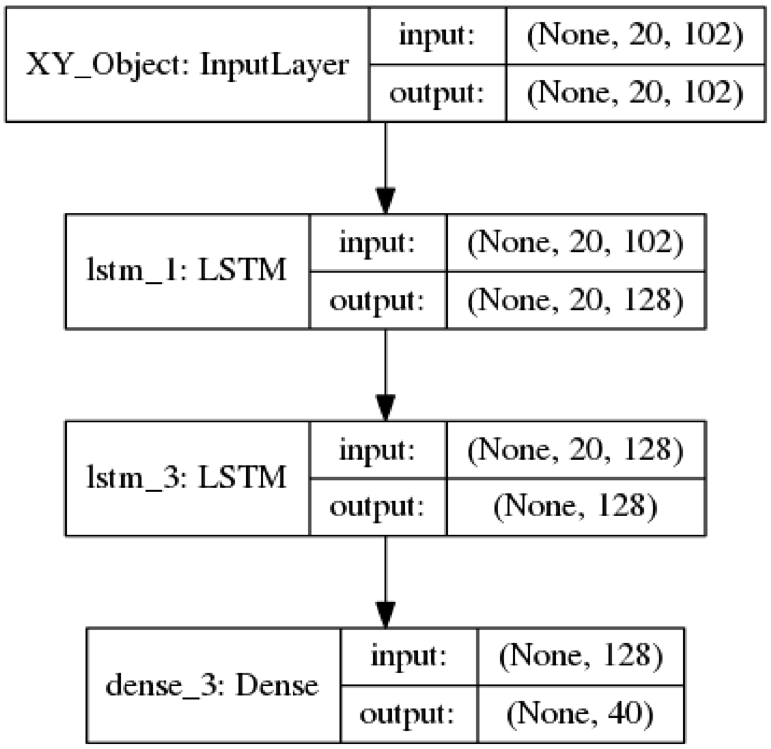


Figure 6.31: Early fusion latent space for x, y and grid map.

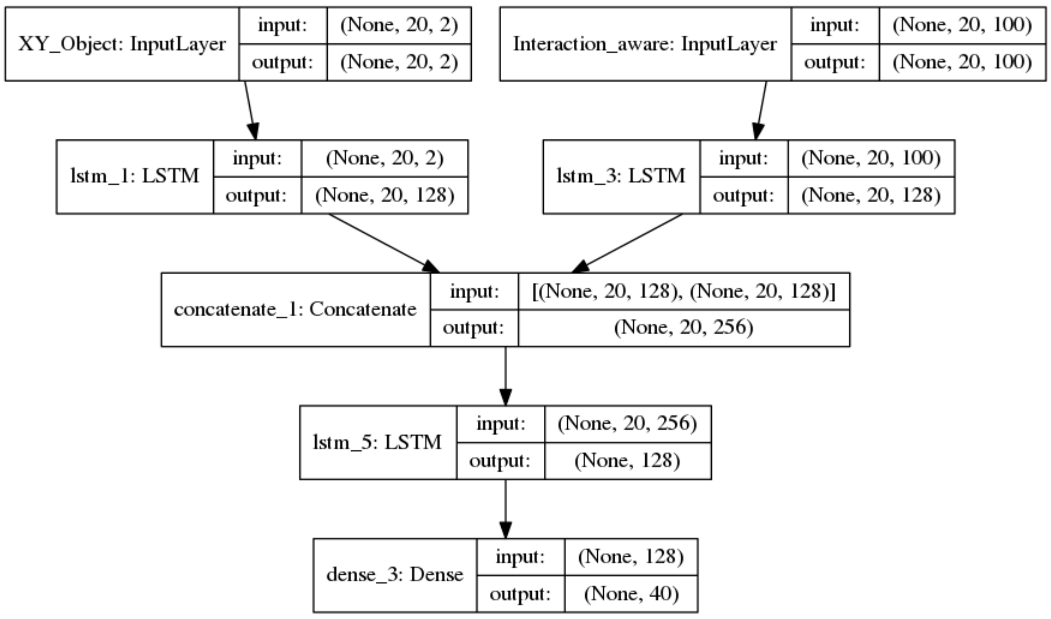


Figure 6.32: Middle fusion latent space for x, y and grid map.

with the x, y positional information of the objects. Similar to the experiments where image features are used, in this set of experiments we first trained two CNN models

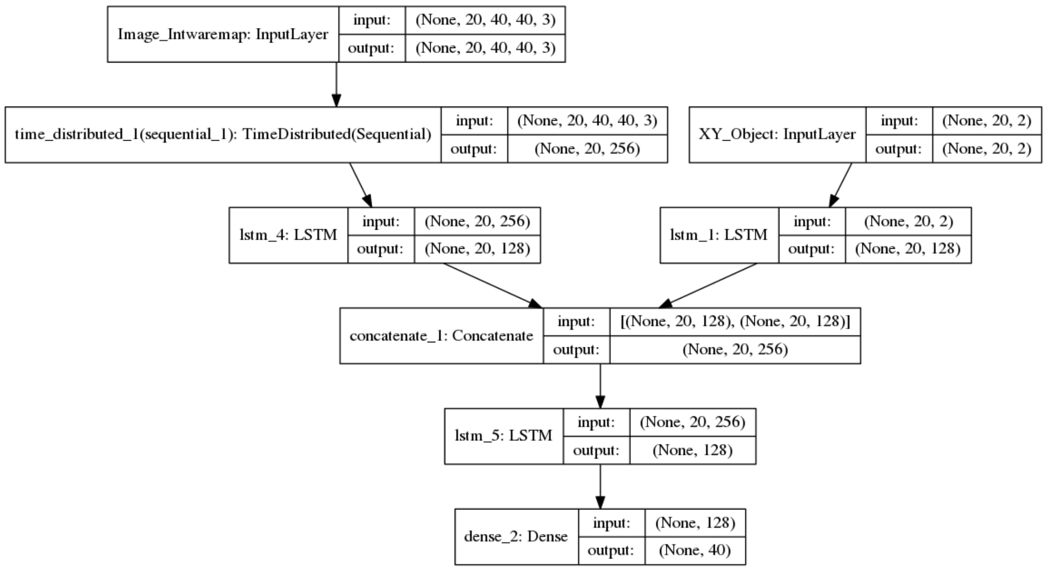


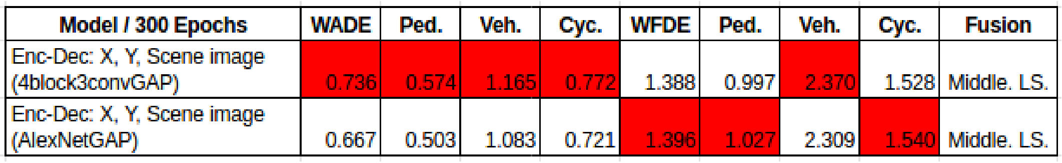
Figure 6.33: Fusion model for x, y and local BEV map image.

for 300 epochs on the k-fold numbers 1, 3 and 5. This way we can select the one that performs better, the two models are listed bellow:

4block3convGAP: this is a four block model, each block with 3 conv layers and a final GAP layer. It can be seen as a deep model.

AlexNetGAP: AlexNet with all layers and a final GAP layer. It can be seen as a shallow but wide model.

The mean results are presented in Figure 6.34. It can be seen that AlexNetGAP has better performance for ADE. However, for FDE 4block3convGAP is better though the difference in error is not significant. Therefore AlexNetGAP was selected as the most promising model.

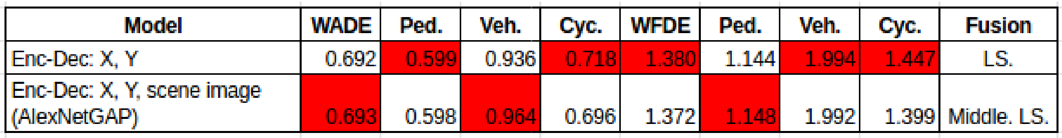


Model / 300 Epochs	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y, Scene image (4block3convGAP)	0.736	0.574	1.165	0.772	1.388	0.997	2.370	1.528	Middle. LS.
Enc-Dec: X, Y, Scene image (AlexNetGAP)	0.667	0.503	1.083	0.721	1.396	1.027	2.309	1.540	Middle. LS.

Figure 6.34: 4block3convGAP vs AlexNetGAP for scene image.

The next step was to train the AlexNetGAP for 1000 epochs on the 5 folds. The results are depicted in Figure 6.35. It can be observed that the use of AlexNetGAP to extract features from scene images leads to a slight error reduction against the baseline model (using only x, y positional information). The improvement is for FDE and mostly for the object cyclist (the worst performing object class).

In this experiment the images were resized to 124×37 pixels and it's possible that an increase in image size could lead to an increase in performance as having bigger images could give more context to the prediction. Another possible improvement could be to feed the model with images already pre-processed by a semantic segmentation model. In this way the model can focus only on high potential areas such as road and sidewalks.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
Enc-Dec: X, Y, scene image (AlexNetGAP)	0.693	0.598	0.964	0.696	1.372	1.148	1.992	1.399	Middle. LS.

Figure 6.35: x, y and scene image: x, y only vs AlexNetGap for scene image.

The Encoder-Decoder model used to fuse x, y positional information and scene image is presented in Figure 6.36.

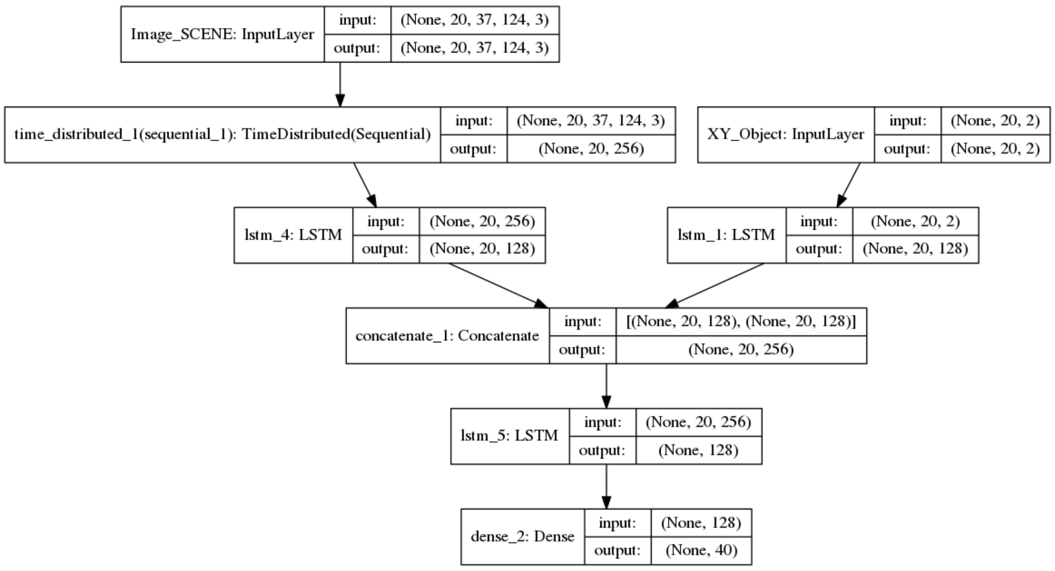
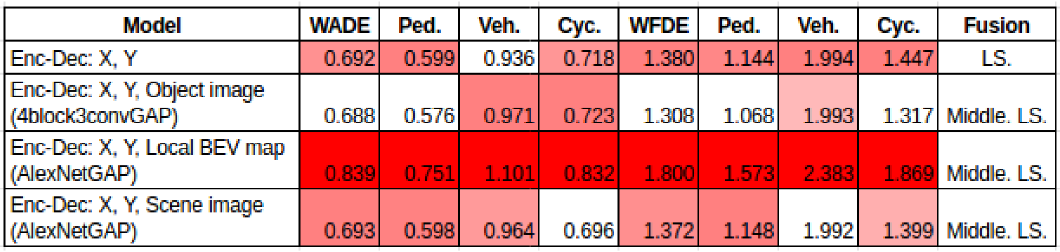

Image_SCENE: InputLayer | input: (None, 20, 37, 124, 3) | output: (None, 20, 37, 124, 3)
time_distributed_1(sequential_1): TimeDistributed(Sequential) | input: (None, 20, 37, 124, 3) | output: (None, 20, 256)
XY_Object: InputLayer | input: (None, 20, 2) | output: (None, 20, 2)
lstm_4: LSTM | input: (None, 20, 256) | output: (None, 20, 128)
lstm_1: LSTM | input: (None, 20, 2) | output: (None, 20, 128)
concatenate_1: Concatenate | input: [(None, 20, 128), (None, 20, 128)] | output: (None, 20, 256)
lstm_5: LSTM | input: (None, 20, 256) | output: (None, 128)
dense_2: Dense | input: (None, 128) | output: (None, 40)

Figure 6.36: Fusion model for x, y and scene image.

6.7.1.6 Comparison of Deep Features

This section presents the performance using deep features in the prediction task. A summary of the results is presented in Figure 6.37. The first row shows performance of the baseline model where it uses only x, y positional information. The following three rows present the results obtained when combining x, y with one of the three types of deep features available – object image, local BEV image map and scene image. The following observations were made:

- The deep features that lead to a better performance when combining with x, y positional information come from the object image (object bounding box).
- The deep features from the whole scene image lead to a slight error reduction for FDE and mostly for the object cyclist.
- The deep features from the local BEV image map does not give any error reduction. This could be due to the fact that KITTI dataset has few crowded scenes where objects interact.
- From the three experiments where deep features were used, something important to point out is the impact that a CNN model has in the overall path prediction task. It would be interesting to evaluate more CNN models and see if some of them with common characteristics such as number of CNN layers, filter size, or specific type of layers (GAP, Attention, etc) perform better in the path prediction task.

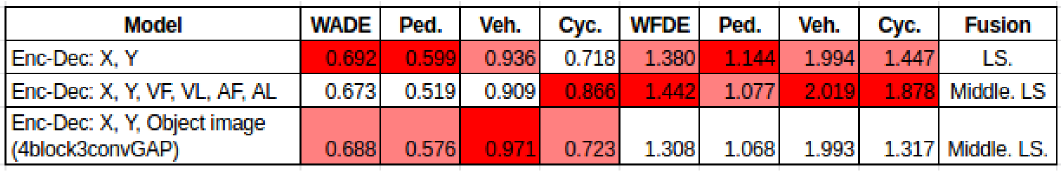


Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
Enc-Dec: X, Y, Object image (4block3convGAP)	0.688	0.576	0.971	0.723	1.308	1.068	1.993	1.317	Middle. LS.
Enc-Dec: X, Y, Local BEV map (AlexNetGAP)	0.839	0.751	1.101	0.832	1.800	1.573	2.383	1.869	Middle. LS.
Enc-Dec: X, Y, Scene image (AlexNetGAP)	0.693	0.598	0.964	0.696	1.372	1.148	1.992	1.399	Middle. LS.

Figure 6.37: Comparison of deep features.

6.7.1.7 Combinations of Features that Lead to a Better Performance

This section presents the combinations of features that improve the performance over the baseline model using only x, y positional information. The Figure 6.38 shows the performance of the baseline model in the first row compared to the two best combination of features. It can be seen that combining the x, y features with the ego-vehicle information (VF, VL, AF, AL) (second row) leads to better performance mostly for ADE metric. The combination of x, y features with Object image also improves the performance against the baseline model for both ADE and FDE. However, the reduction in error is greater for FDE.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
Enc-Dec: X, Y, VF, VL, AF, AL	0.673	0.519	0.909	0.866	1.442	1.077	2.019	1.878	Middle. LS
Enc-Dec: X, Y, Object image (4block3convGAP)	0.688	0.576	0.971	0.723	1.308	1.068	1.993	1.317	Middle. LS.

Figure 6.38: Combinations of features that lead to a better performance.

6.7.1.8 X,Y , Object Image, Ego-vehicle Features

From the previous experiments where two types of features were combined, it can be observed that ego-vehicle information and visual information of the object lead to an error reduction in the path prediction task. So in this experiment those two features are combined with the x, y positional information in an end-to-end architecture to see if further improvement can be reached. The model architecture is shown in the Figure 6.39

The results are presented in Figure 6.40. Comparing the result of this last combination against the baseline model (row 1), it is clear that no improvement is achieved. One reason for this could be that as the model complexity is higher, because more features are now being fused, more epochs are required to achieve better results.

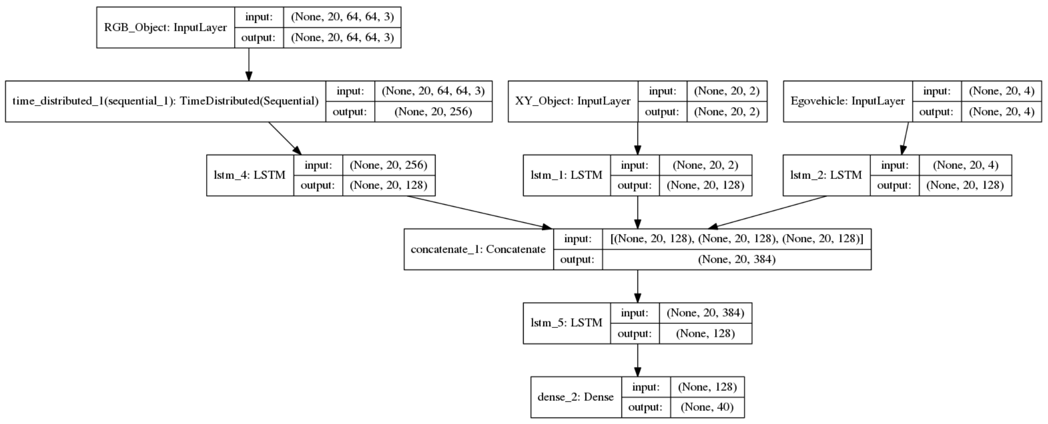
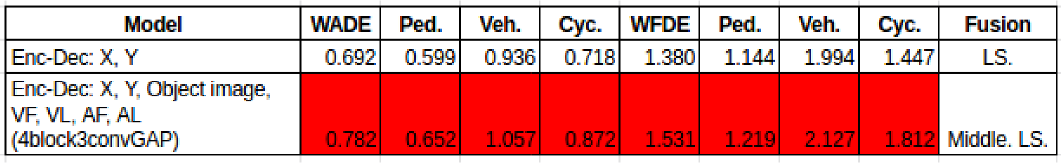


Figure 6.39: Model architecture of Combination of three features: x, y , object image, and ego-vehicle information.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
Enc-Dec: X, Y, Object image, VF, VL, AF, AL (4block3convGAP)	0.782	0.652	1.057	0.872	1.531	1.219	2.127	1.812	Middle. LS.

Figure 6.40: Combination of three features: x, y , object image, and Ego-vehicle information.

6.7.1.9 Impact of Fusion Strategies

The way in which the available features are combined in a model is called fusion strategies and it is still an open challenge. In this chapter, the impact of the fusion strategies was evident when evaluating three combinations of features. The first initial evidence was presented in **X,Y features** in the Figure 6.16, the results showing that representing the features in a latent space improves the performance compared to using the features directly to the LSTM used. The second evidence was presented in **X,Y, Ego-vehicle features**, the results in the Figure 6.19 indicate that middle fusion of features represented in latent space got better performance, on the contrary early fusion of raw features led to higher error in path prediction. The third evidence was presented in **X,Y, Interaction-aware map**, the results exhibited in the Figure 6.27 revealed that there is error reduction when using middle fusion of features in the latent space. Early fusion however leads to a higher error. This is observed for both grid and polar map.

Looking at the impact of the evaluated fusion strategies, it would be interesting to explore more in this area, since not only the features are important but also the way in which these features are joined in a model architecture.

6.7.2 Discussion of Exploring Multimodal Features

Exploring multimodal features to improve the path prediction task was an entertaining journey that gave interesting results. The following observations can be drawn:

- **Latent space representation**. The first interesting observation to point out is that representing the raw features first in a latent space improves the performance compared with using the raw features directly to feed an LSTM. This was clearly seen in most of the combination of features.

- **Fusion strategies**. It can also be noted that the way in which the features are fused in the model architecture has a significant impact on the prediction. This means that, besides paying attention to the features used, it is also recommended to explore different fusion strategies to combine the chosen features. The best performing fusion strategy was middle fusion of features in latent space. Using directly the raw features led to higher error.

- **CNN models**. Choice and configuration of CNN models to extract features from images have an impact in the overall model, so when dealing with images an evaluation of different CNN models is desirable.

- **Best combination of features**. It can be seen that combining the x, y features with the ego-vehicle information (VF, VL, AF, AL) leads to better performance mostly for ADE metric. The combination of x, y features with Object image also improves the performance against the baseline model for both ADE and FDE. However, the reduction in error is greater for FDE.

- The use of interaction-aware features did not lead to any improvement over the baseline method, this could be due to the fact that KITTI dataset has few crowded scenes where the objects interact with each other.

6.8 Ensembles

Ensemble building is a common way to improve the performance of individual models and an ensemble of individual predictor can outperform a single predictor in the average [15]. Ensemble methods, which work on a higher level to improve the performance of "unstable" predictors such as decision tree and neural networks, have been successfully employed for solving pattern classification, regression, time series forecasting and fault prediction problems [42].

There are several ways of building ensembles. It could be by varying the choice of the data used to train each model in the ensemble (e.g., bootstrap aggregation or bagging), varying the choice of the models used, or by varying the way in which the outcomes from each model in the ensemble are combined.

In this section, looking at the models showed greatest improvement in performance compared to our baseline methodology, the following ensembles were built:

- Ensemble 1: [Enc-Dec: X, Y], [Enc-Dec: X,Y, VF, VL, AF, AL]. This groups the baseline model with the model combining x, y features and ego-vehicle information. Figure 6.41 details the architecture of this ensemble.
- Ensemble 2: [Enc-Dec: X, Y], [Enc-Dec: X,Y, Object image (4block3convGAP)]. The second ensemble constitutes the baseline model and the model that combines x, y features and the object image. The architecture of this ensemble is illustrated in the Figure 6.42.
- Ensemble 3: [Enc-Dec: X, Y], [Enc-Dec: X,Y, VF, VL, AF, AL], [Enc-Dec: X,Y, Object image (4block3convGAP)]. The third ensemble uses three models: 1. Baseline model; 2. Model combining x, y features and ego-vehicle

information; 3. Model which combines x, y features and the object image. The Figure 6.43 presents the architecture of this ensemble.

As shown in Figure 6.41, Figure 6.42, and Figure 6.43, each ensemble uses different features, different models and the output of each model is combined in the ensemble by averaging in the final output layer.

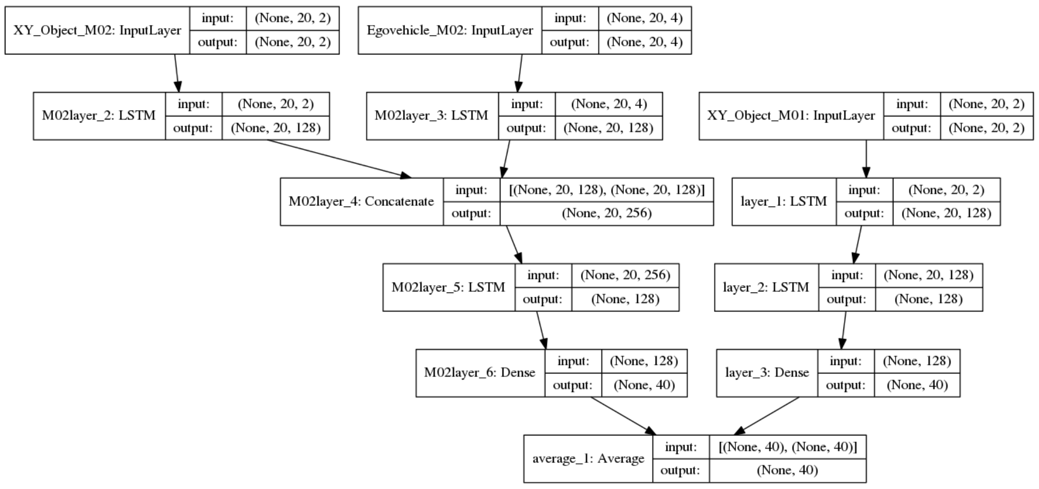

XY_Object_M02: InputLayer | input: (None, 20, 2) | output: (None, 20, 2)
Egovehicle_M02: InputLayer | input: (None, 20, 4) | output: (None, 20, 4)
M02layer_2: LSTM | input: (None, 20, 2) | output: (None, 20, 128)
M02layer_3: LSTM | input: (None, 20, 4) | output: (None, 20, 128)
XY_Object_M01: InputLayer | input: (None, 20, 2) | output: (None, 20, 2)
M02layer_4: Concatenate | input: [(None, 20, 128), (None, 20, 128)] | output: (None, 20, 256)
layer_1: LSTM | input: (None, 20, 2) | output: (None, 20, 128)
M02layer_5: LSTM | input: (None, 20, 256) | output: (None, 128)
layer_2: LSTM | input: (None, 20, 128) | output: (None, 128)
M02layer_6: Dense | input: (None, 128) | output: (None, 40)
layer_3: Dense | input: (None, 128) | output: (None, 40)
average_1: Average | input: [(None, 40), (None, 40)] | output: (None, 40)

Figure 6.41: Ensemble 01: x, y only with x, y and ego-vehicle data.

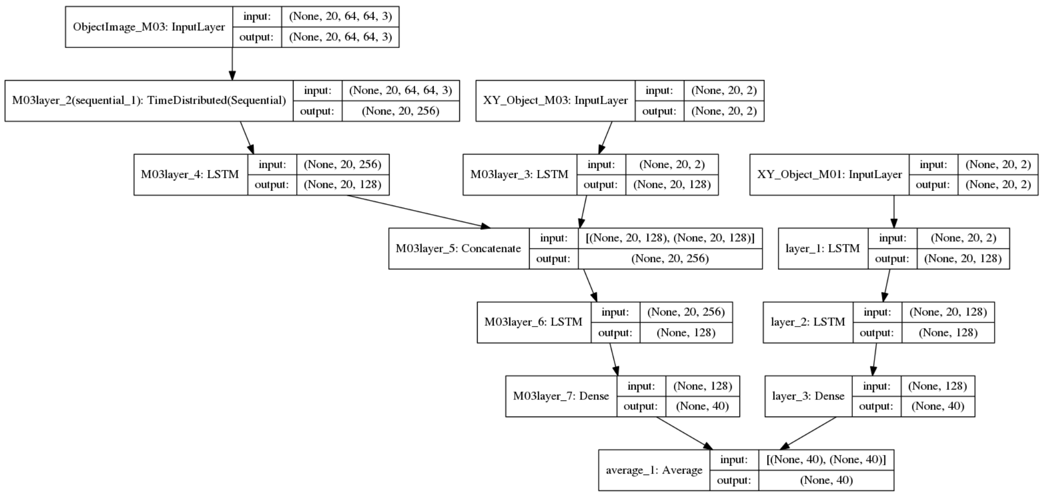

ObjectImage_M03: InputLayer | input: (None, 20, 64, 64, 3) | output: (None, 20, 64, 64, 3)
M03layer_2(sequential_1): TimeDistributed(Sequential) | input: (None, 20, 64, 64, 3) | output: (None, 20, 256)
XY_Object_M03: InputLayer | input: (None, 20, 2) | output: (None, 20, 2)
M03layer_4: LSTM | input: (None, 20, 256) | output: (None, 20, 128)
M03layer_3: LSTM | input: (None, 20, 2) | output: (None, 20, 128)
XY_Object_M01: InputLayer | input: (None, 20, 2) | output: (None, 20, 2)
M03layer_5: Concatenate | input: [(None, 20, 128), (None, 20, 128)] | output: (None, 20, 256)
layer_1: LSTM | input: (None, 20, 2) | output: (None, 20, 128)
M03layer_6: LSTM | input: (None, 20, 256) | output: (None, 128)
layer_2: LSTM | input: (None, 20, 128) | output: (None, 128)
M03layer_7: Dense | input: (None, 128) | output: (None, 40)
layer_3: Dense | input: (None, 128) | output: (None, 40)
average_1: Average | input: [(None, 40), (None, 40)] | output: (None, 40)

Figure 6.42: Ensemble 02: x, y only with x, y and object image.

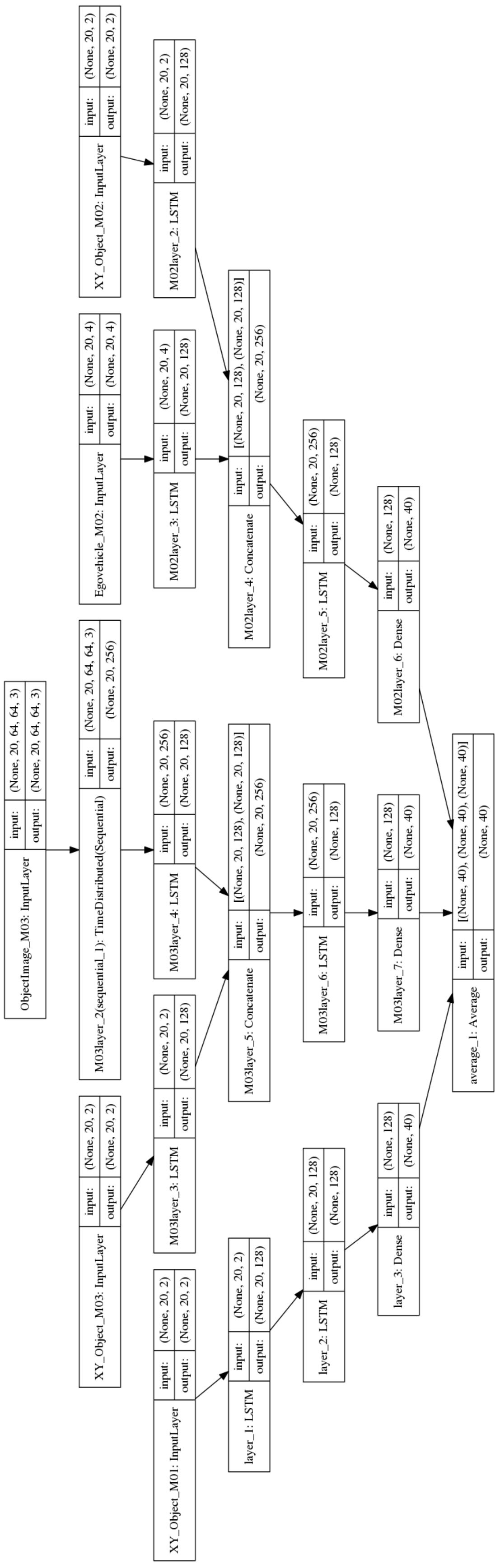
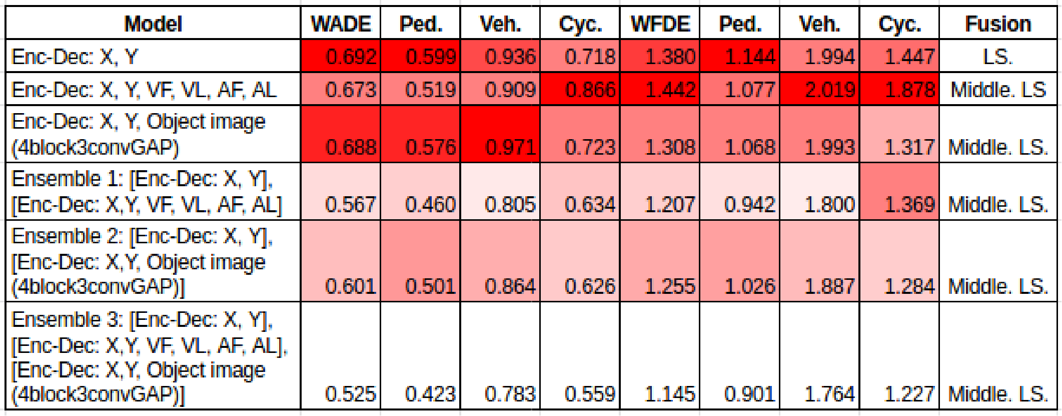


Figure 6.43: Ensemble 03: x, y only with x, y and ego-vehicle data with x, y and object image.

The results of each ensemble are presented in the Figure 6.44. It can be observed that the three ensembles reach better performance than the baseline model (row 1). The three ensembles also reach better performance that the two individual combinations of features (row 2 and 3). The ensemble with better results is the Ensemble 3 which leverages three models, the baseline model and the two best models that use ego-vehicle features and deep features from object image.



Model	WADE	Ped.	Veh.	Cyc.	WFDE	Ped.	Veh.	Cyc.	Fusion
Enc-Dec: X, Y	0.692	0.599	0.936	0.718	1.380	1.144	1.994	1.447	LS.
Enc-Dec: X, Y, VF, VL, AF, AL	0.673	0.519	0.909	0.866	1.442	1.077	2.019	1.878	Middle. LS
Enc-Dec: X, Y, Object image (4block3convGAP)	0.688	0.576	0.971	0.723	1.308	1.068	1.993	1.317	Middle. LS.
Ensemble 1: [Enc-Dec: X, Y], [Enc-Dec: X,Y, VF, VL, AF, AL]	0.567	0.460	0.805	0.634	1.207	0.942	1.800	1.369	Middle. LS.
Ensemble 2: [Enc-Dec: X, Y], [Enc-Dec: X,Y, Object image (4block3convGAP)]	0.601	0.501	0.864	0.626	1.255	1.026	1.887	1.284	Middle. LS.
Ensemble 3: [Enc-Dec: X, Y], [Enc-Dec: X,Y, VF, VL, AF, AL], [Enc-Dec: X,Y, Object image (4block3convGAP)]	0.525	0.423	0.783	0.559	1.145	0.901	1.764	1.227	Middle. LS.

Figure 6.44: Ensembles results.

Finally, Figure 6.45 presents the mean results of the best models and their standard deviation as error bars. In general, for the weighted metrics, WSADE and WSFDE, the results obtained in each K fold for the 2 best models and the three ensembles seems to be close to the mean. For the metric WSDE and WSFDE the SD is low which means that the performance of each model in each Kfold is clustered around the mean. This means that the models keep their performance along the 5 folds. Looking individually per object type, the performance of all the models are more dispersed from the mean for the object vehicle and cyclist specifically for the metric FDE. This means that for that type of objects and metric the performance of the models are not constant along the 5 folds. This means that the models are fragile for the object Vehicle and Cyclist when predicting further ahead which is captured by the Final Displacement Error (FDE) metric.

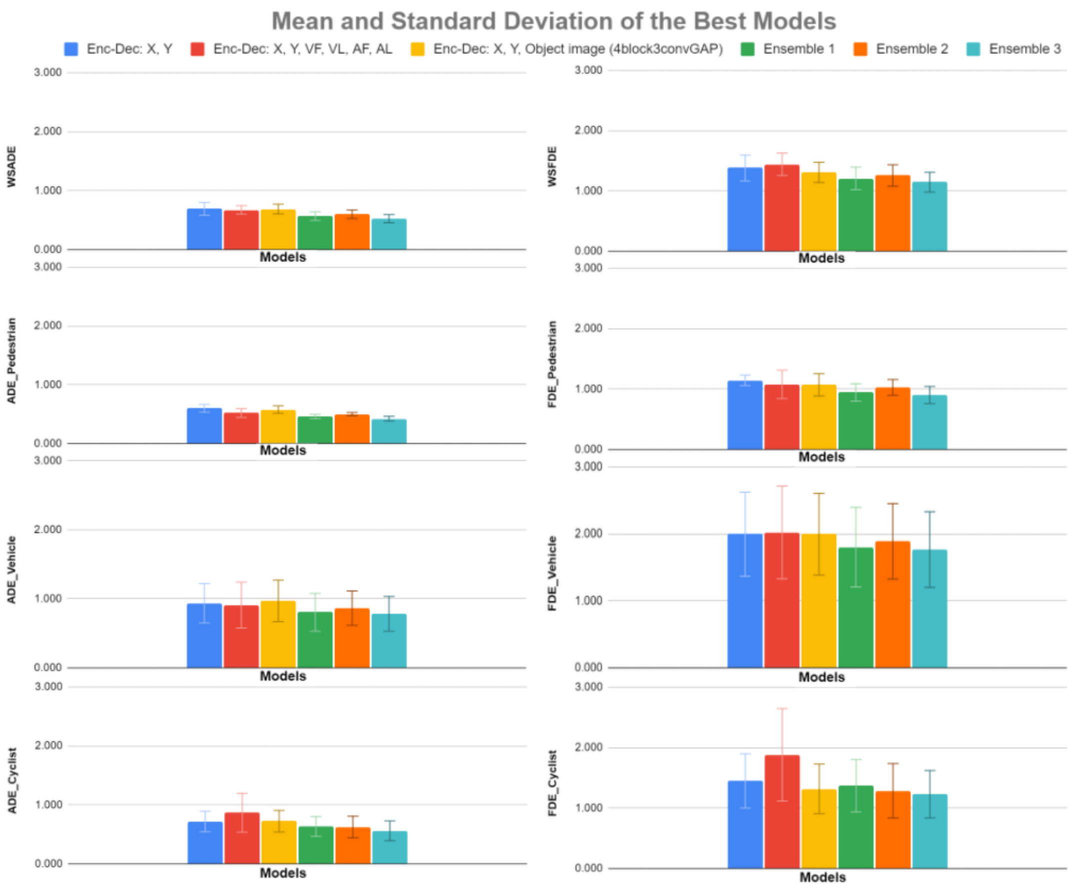


Figure 6.45: Mean and Standard Deviation of the best models.

6.9 Discussion and Conclusions

This chapter presents the exploration of using contextual information in the path prediction task. Several features were evaluated to see if they help to reduce the prediction error. Additionally some different fusion strategies were evaluated to see which performs better when using the different types of features. Also, some CNN architectures were explored to extract features of images. All this was put together in a end-to-and architecture. Finally, an exploration of basic ensembles was carried out to see if any improvement can be obtained on the performance of individual models. Observing the obtained results, several conclusions can be drawn.

LSTMs for sequential and enriched trajectory information. The initial exploration of LSTM architectures was done in chapter 3, where a single-shot approach was developed to process x, y positional information only. In this chapter, a deeper exploration was done by adding more contextual features to the tracklets in an end-to-end architecture. The results showed that LSTMs are able to process sequences of enriched trajectory information and in some cases improve the performance by using these enriched trajectories.

Use of contextual information of a scene to improve path prediction results. This point was deeply explored in chapter 6 and not only several features that describe a vehicle and its surrounds were evaluated but also combined using different fusion strategies. Starting from using only x, y positional information to using features extracted from images, different combinations were done. One interesting observation here is that both the information available and the method of fusing that information is highly important. This could be observed in each combination, where using middle fusion on latent space leads to better performance. To extract features from images CNN models were used. This chapter also showed that the CNN used to extract features has an impact on the prediction results. From here, it can be said that, besides paying attention to the features used, it is also recommended to explore different fusion strategies to combine the used features. Also, if it is the case of using deep features, an evaluation of different CNN models is desirable.

Use of ego-vehicle information to improve path prediction. In this chapter, ego-vehicle features were added to the positional information x, y. The features used were orientation, velocity in [x,y,z] and acceleration in [x,y,z]. A combination of them was evaluated. The results in section 6.6 showed that the ego-vehicle features does not improve the performance of the model for short P.H. but they do for longer P.H. Also, the results in section 6.7 showed that the combination of [X,Y,VF,VL,AF,AL] leads to better performance for the objects pedestrian and vehicle for ADE as shown in the subsection **X,Y, Ego-vehicle features** and **Combinations of features that lead to a better performance**. The best fusion strategy was middle fusion of features in latent space. Using directly the raw features leads to higher error (presented in (X,Y, Ego-vehicle features)).

Ensembles. Interesting results were achieved by the ensemble methods. It is known that ensembles improve the performance of individual classifiers or methods, and in the section 6.8 this was evident in the context of path prediction in traffic scenes. The three ensembles that were built in this final section improved the performance of the baseline methodology and the two other best individual models. The ensemble with the best results is the Ensemble 3: [Enc-Dec: X, Y], [Enc-Dec: X,Y, VF, VL, AF, AL], [Enc-Dec: X,Y, Object image (4block3convGAP)], which includes the baseline methodology and the two best individual models.

Processing time. Using a computer with the following features: GPU GeForce GTX 980, CPU Intel® Core^{TM} i5-4690K CPU @ 3.50GHz x 4, RAM 24GB. The processing time per epoch using the 4block3convGAP model was 137s for object image, 97s for interaction-aware image, and 166s for scene image. Using the AlexNetGAP model the processing time per epoch was 97s, 75s, and 140s for object, interaction-aware and scene image respectively. During inference, the average processing time per tracklet is 21.069 ms/tr with 4block3convGAP model and 20.564 ms/tr with the model AlexNetGAP. That compared with time taken by the baseline model that was 1.859 ms/tr. Finally, the processing time for each ensemble during inference was, 25.327 ms/tr, 29.151 ms/tr , and 42.990 ms/tr for the ensemble 1, 2 and 3 respectively. All this for prediction time horizon (P.H.) of ± 20. It can be noticed that using ensembles increases the processing time during inference. Of course, the processing time depends on the size of the images used. In this work images of size 64×64, 40×40 and 124×37 pixels were used for object image, interaction-aware local BEV map image, and scene image.

Chapter 7

Conclusion

7.1 Thesis Overview

In this research thesis, the aim was to analyse the problem of path prediction in traffic scenes, specifically in the context of traffic scenes from ego cameras such as those mounted on a vehicle, with the objective of exploring existing techniques and improving state-of-the-art methods in this research area. In existing works, path prediction has been addressed as a regression problem, looking at a path as time series data and the prediction of a future path is based on the previously observed path.

A significant outcome of this thesis is on the evaluation of LSTMs on the path prediction task and the fusion of features. Starting from constructing path prediction information as a time series data. Different variants of LSTMs were evaluated specifically in this task to understand their behaviour and performance. Also, as LSTMs have the limitation of only predicting one path per observed input tracklet, LSMTs were extended to predict multiple paths with associated uncertainty by combining them with a Mixture Density Network (MDN) as a final layer. Then, a deep exploration was done on combining different types of contextual information with the normally used positional information of objects. During the exploration of this contextual information different fusion strategies were evaluated showing interesting results. In addition, as deep features were used in some models, the impact of CNNs were analysed in the context of path prediction. Finally, to conclude this research thesis journey, ensemble models were investigated.

A summary of the investigations, research findings, experimental results, and the proposed solutions in this thesis is as follows.

Chapter 1 presents the motivation and importance of tackling the problem of path prediction in traffic scenes along with a brief overview of the challenge. This chapter also presents the main objective of this research and lists the hypotheses and research questions that are the basis of this thesis.

Chapter 2 defines the important concepts required to understand the context of this research followed by the analysis and classification of related works that constitute the state of the art in the area of path prediction. The available datasets and evaluation metrics used for evaluation are also discussed in this chapter.

From the literature review, it was observed that significant research has been done on path prediction in general, mostly using static cameras, but work specifically in the context of cameras mounted on vehicles, such as those used in modern ADAS, still requires work to be done. Some interesting points were drawn from the literature review:

- Several approaches were documented in the analysed works, ranging from the well known Kalman filter through to clustering techniques and more recently to deep neural networks. From here two main approaches were identified, The Kalman Filter (KF) and the LSTMs (Long Short-Term Memory Networks).
- There is trend to use more features along with the x, y positional information of the object that should be considered when predicting a path.
- No specific way of representing and fusing additional features in this context was identified – we describe this as an "enriched tracklet". What could be noted is that deep neural networks are used as a means of fusing different types of information in other contexts.

Regarding the analysis of the available datasets, most of the works use datasets where labeled data was not available so they have to also tackle the problem of

Object Detection and Tracking. KITTI was the dataset that included labeled data of detected and tracked objects along with more data of the ego-vehicle and the scene so this dataset was adopted. However, this data was not specifically annotated for path prediction. Recently some datasets such as ApolloScape [109], nuScenes [115], Lift [108, 118], and Argoverse [102], started to provide data specifically for the task of path prediction along with the standardisation of two metrics to rate performance, Average Displacement Error (ADE) and Final Displacement Error (FDE). Even though those datasets were not used, the two metrics were adopted.

Chapter 3 From the literature review, two main approaches were identified, the Kalman Filter (KF) and LSTMs (Long Short-Term Memory Networks). The objective of this chapter was to evaluate the performance of a baseline LSTM and the KF to predict the future position of objects that are normally present in traffic scenarios, such as pedestrians, vehicles and cyclists, for different time prediction horizons using the KITTI dataset. The data definition is also given in this chapter to construct the data as time series.

This chapter presents a single-shot prediction approach, which consists of using an LSTM in a multiple output strategy, developing this way a model to predict an entire sequence in a one-shot manner.

The following conclusions were reached in this chapter:

1. LSTM vs KF: The results show that LSTM Relative Tracklet Position (RTP) outperforms the simple Kalman Filter in most cases, specifically for vehicle and pedestrians. An LSTM achieves good performance for a P.H. ± 5 of up to an ADE of 0.01m for pedestrians, 0.06m for vehicles and 0.02m for cyclists and up to a FDE of 0.016m, 0.15m, 0.04m for the same objects improving the performance of the baseline Kalman Filter with an ADE of 0.06m 0.46m 0.23m and FDE of 0.07m 0.57m 0.27m for pedestrians, vehicles and cyclists respectively.

2. LSTM Relative Tracklet Position (RTP) VS LSTM Absolute Tracklet Position

(ATP): This set of results show clearly that there was error reduction in most cases when translating the tracklets to relative position (RTP). In a few cases, mostly pedestrians, a slight increase in error was found.

3. Long P.H.: As expected, the results also indicated that the performance is affected by the prediction horizon where the longer the prediction horizon, the bigger the displacement error. The prediction horizon where the approach is most reliable is for ± 5 and ± 10 for image coordinates and for ± 5 to ± 15 for birds-eye view perspective.

4. Birds-eye view vs image coordinates: the approach performs better when predicting in birds-eye view, since this is a measurement of the real world (metres), compared to using image coordinates (pixels). However, this perspective is not always available when using only standard cameras.

Chapter 4 Considering the performance of the Vanilla LSTMs in the previous chapter and looking at the literature about LSTMs and how this architecture has been applied, the objective of this chapter was to compare the performance of some models against the baseline LSTM Vanilla model to predict the future position of objects in traffic scenes. The dataset used for evaluating the models was KITTI. As in chapter 3, the models were evaluated on three objects – pedestrians, vehicles, and cyclists, for four prediction horizons (P.H.) from ± 5 to ± 20 steps and only on the BEV perspective. The objective of each model was also the same as chapter 3 which is to predict a single path, tr^P, based on the past observed tracks of one object, tr^O.

The following conclusions were drawn:

1. Best model: non of the models perform better in all the cases. GRU seems to be one the model that performs better in most of the cases. However, Vanilla LSTM have an average performance compared with the other models.

2. P.H: the performance of all the models decrease when predicting for long time

prediction horizons (P.H). Some models such as LSTM Encoder-Decocer and the one using 1D convolution are better for predicting long time P.H which worth further work.

3. Type of object: Considering the type of object, all the models appear to struggle when predicting trajectories for the object vehicle, followed by the object cyclist and for the object pedestrian.

4. Finally, different to the studies shown in the introduction of this chapter, in this experiments the improvement in performance of some models against the vanilla LSTM is significant and this significance increase when longer the time prediction horizon is. However, it is important to say that non of the models perform better than the vanilla LSTM in all the cases. This means that Vanilla is still a good choice.

Chapter 5 As shown in the previous chapters, path prediction using LSTMs has shown good performance. However, most of the approaches are limited to only predicting a single path per observed tracklet. Path prediction is better structured as a non-deterministic task and requires predicting with a level of uncertainty. In addition, generating a set of paths instead of a single one is a more realistic manner of predicting the possible position of objects, since a set of paths can be used more extensively for both route optimisation and object avoidance applications. Because of that, this chapter presented an approach that allows for predicting a set of paths with associated uncertainty per observed tracklet.

This approach uses an LSTM with a MDN layer which is called Mixture Density Models (MDMs) and was compared against two baselines methodologies – The Kalman Filter (KF) with Constant Velocity (CV) model and a Vanilla LSTM, to establish that our approach does not lose accuracy when predicting a set of paths. In addition, an initial exploration of including more features to the tracklets was done to see if this lead to any improvement. The datasets used for evaluation were KITTI and CityFlow. As in chapter III, the models were evaluated on three objects –pedestrian, vehicles, and cyclists for KITTI and on vehicles for CityFlow. For four prediction horizons (P.H.) from ± 5 to ± 20 steps and RTP data was used. Finally, the approach was also evaluated on the generation of different numbers of paths, two to five. From the experiments the following observation were noted:

1. Overall performance: LSTMs combined with MDM do not decrease the performance, in some cases the approach even got better results than the two baseline methods.

2. Use of extra features: The experiments showed that the extra features lead to better results overall. This was evidenced in KITTI for short prediction horizons for the object pedestrian and vehicle and in CityFlow for the object vehicle for P.H. of less than ± 15.

3. Relationship between the accuracy of the predicted set of paths and its probability: it was evidenced that the predicted paths are more similar to the ground truth when the component that is predicting them has high probability. However, paths that are predicted for the components with low probability are increasingly different to the ground truth. This conclusion is desirable when predicting paths, because based on the probability of each predicted path we can relay more in those paths with high probability.

4. Inconsistency when predicting more than three paths per input tracklet: the approach was also evaluated for predicting multiple numbers of paths per input tracklet. It was observed that when predicting two to three paths per input, the approach works well as the predicted paths are still related to the ground truth. However, in some cases, when predicting four and five paths, some of the predicted paths begin to deviate further from the ground truth. This cannot be seen as a disadvantage since each path has a probability, so by looking at the probability of each path, those paths with very low probability can be discarded.

5. Finally, it was shown that LSTMs can be extended from predicting a single path per input tracklet to predict multiple paths with associated uncertainty by combining it with a Mixture Density Network (MDN) layer. However, proper understanding of the output parameters is required.

Chapter 6 examines the use of contextual information in the path prediction task. Several features were evaluated to see if they reduce the error when predicting. In addition, different fusion strategies were evaluated to assess performance when using the different type of features. With the constraints of available GPU resources, some CNN architectures were explored to extract features of images. These approaches were finally amalgamated in a end-to-end architecture and the features that best contributed to reducing the prediction error were identified together with the fusion strategy that gives better results.

- **LSTMs for sequential and enriched trajectory information**. The initial exploration of LSTM architectures was done in chapter 3, where a single-shoot approach was developed to process x, y positional information only. In this chapter 6 a deeper exploration was done on adding more contextual features to the tracklets by an end-to-end architecture. The results evidenced that LSTMs are able to process sequence of enriched trajectory information and in some cases improve the performance by using this enriched trajectories.

- **Use of contextual information of a scene to improve path prediction results**. This point was deeply explored in chapter 6 and not only several features that surrounds a vehicle were evaluated but different fusion strategies. Starting from using only x, y positional information to use features extracted from images, different combinations were done. One interesting observation here is that both, the information available and the way of fusing that information is highly important. This could be observed in each combination, where using middle fusion on latent space leads to better performance. To extract features from images CNN models were used. In this chapter was also evidenced that the CNN used to extract features have an impact on the prediction results. From here, it can be said that, besides paying attention to the features used, it is also recommended to explore different fusion strategies to combine the used features. Also, if it is the case of using deep features, an evaluation of different CNNs models is desirable.

- **Use of ego-vehicle information to improve path prediction**. In this chapter 6, ego-vehicle features was added to the positional information x, y. The features used were orientation, velocity in [x,y,z], acceleration in [x,y,z]. A combination of them was evaluated. The results in section **6.6 Exploration of ego-vehicle features** shown that the ego-vehicle features do not improve the performance of the model for short P.H. but they do for longer P.H. Also, the results in section **6.7 Exploration of multimodal features** shown that the combination of [X,Y,VF,VL,AF,AL] leads to better performance for the object pedestrian and vehicle for ADE as shown in the subsection **X,Y, Ego-vehicle features** and **Combinations of features that lead to a better performance**. The best fusion strategy was middle fusion of features in latent space. Using directly the raw features leads to higher error (presented in **X,Y, Ego-vehicle features**).

- It is important to point out that representing first the raw features in a latent space improve the performance against using the raw features directly to feed an LSTM. This was clearly seen in most of the combination of features. It was also observed that the way in which the features are fused in the model architecture has a significant impact in the prediction. The best fusion strategy was middle fusion of features in latent space. Using directly the raw features lead to higher error. In addition, the combinations of features that lead to better performance were 1) x, y, ego-vehicle (VF, VL, AF, AL) features and 2) x, y, Object image features.

- **Ensembles**. To conclude, interesting results were gotten by ensemble methods, it is known that ensembles improve the performance of individual classifiers or methods, and in the sections **6.8 Ensembles** this was evidenced in the context of path prediction in traffic scenes. The three ensembles that were built in this final section improve the performance of the baseline methodology and the two other best individual models. The ensemble with the best results is the Ensemble 3: [Enc-Dec: X, Y], [Enc-Dec: X,Y, VF, VL, AF, AL], [Enc-Dec: X,Y, Object image (4block3convGAP)], which includes the baseline methodology and the two best individual models.

7.2 Research Questions and Proposed Solutions

This thesis focuses on the evaluation of LSTM architectures for path prediction and that its performance can be improved by using contextual information and extended to predict multiple paths. The hypothesis and research questions are reviewed and analysed according to the the experimental results.

Hypothesis: *LSTMs are an effective tool for path prediction and existing work can be extended to predict multiple paths and to include contextual information, creating a holistic approach leading to improved performance in terms of ADE and FDE*

To investigate the hypothesis, the following questions were explored as follows:

- **Q1 How should the observed object position (tracklets) be best represented?** In this research, the prediction of the future path of a moving object is based on its past observed path (tracklets) along with more information of the scene (context) where this prediction is happening. This first research question aims to investigate how positional information can be best represented to be fed to our LSTMs-Based approach. RQ1 question was analysed in chapter 2, and explored and evaluated in chapter 3. From chapter 2 and 3 it was evidenced that representing the position of objects as time series and as Relative Tracklet Position (RTP) was suitable to predict the future path of an object. Furthermore, in chapter 6 was concluded from section 6.7 Exploration of multimodal features that representing the object position first in a latent space led to error reduction.

- **Q2 How can LSTMs be extended to predict multiple paths?** Predicting a set of paths with associated uncertainty is a more realistic way of predicting the future position of objects instead of a single one. LSTMs are only able of predicting a sigle path per observed tracklet, because of that this research question aims to explore a way to extend LSTMs to predict a set of paths. RQ2 is initially explored in chapter 4, where a study of several variants of LSTMS was performed on predicting the future path of objects in traffic scenes, this with the objective of understanding their behaviour and the way in which each variant processes the input sequential data. RQ2 was also studied in chapter 5 where LSTM architectures are successfully used along with MDN (Mixture Density Networks) for predicting a set of paths per observed tracklet along with its associated uncertainty. Chapter 5 evidenced that combining LSTMs with a Mixture Density Networks (MDN) as a final layer does not reduce overall performance and, in some cases, the accuracy was improved. It was also evidenced that this approach performs well for predicting up to three paths per input tracklet. However, in some cases, when predicting four and five paths, some of the predicted paths begin to deviate further from the ground truth. This can be solved by looking at their probabilities, the path with low probability can be discarded.

- **Q3 Are Long Short-Term Memory (LSTM) architectures suitable for sequential and enriched trajectory information?** LSTMs have shown good performance when dealing with sequential information such as time series. However, this research aims to include contextual information such as visual features, ego-vehicle information, other objects position leading this way to a holistic approach. The purpose of this RQ3 is to evidence that LSTMs are able to process all these type of information. The initial exploration of LSTM architectures was done in chapter 3, where a single-shot approach was developed using object position only. RQ3 is explored in chapter 5 and more deeply in chapter 6. In chapter 5, an initial exploration of adding more features to the tracklets was done showing that these extra features improved the overall performance of the model. Finally, in chapter 6, a more complete exploration was done by adding additional contextual features to the tracklets. Starting from only using object position to using visual-object (images) features, ego-vehicle information, other object position, and scene-image features different combinations were evaluated. The results showed that LSTMs models are able to process sequences of enriched trajectory information.

- **Q4 How can contextual information of a scene be used to improve path prediction results over only using x, y positional features?** How to process/fuse the available features inside a model is an important point when trying to use different type of information. The fact that a set of features does not lead error reduction does not mean this set is not working, the reason could be the way this set of features are being fused inside a model. This research question aims to find a way to best fuse contextual features for the path prediction task. RQ4 was deeply explored in chapter 6 where additional features describing the environment were included using different fusion strategies. Three fusion strategies were evaluated – Early fusion of raw features, Early fusion of latent space features, and Middle fusion of latent space features. An interesting observation here is that both the information available and the fusion method are highly important. This was observed in each combination where using middle fusion on latent space features leads to better performance.

  The experiments in chapter 6 also showed that the CNN used to extract features can have an impact on the prediction results. This means that the overall model relay on the deep features extracted from images. So attention should be put as well on the selection of a specific CNN.

7.3 Research Contributions

The novel contributions of this work are:

1. **Thorough evaluation of different variants of LSTM models to understand their behaviour and performance on the task of predicting the future path of three different objects – pedestrians, vehicles and cyclists on four prediction horizons.** The mentioned architectures were selected because they process the input data in different manners that a vanilla LSTM. This study allows us to understand better how to use LSTM architectures in its different ways, since this was needed to build more complex model for predicting multiple paths and for processing multimodal data.

2. **Proposing the use of LSTMs with MDN to predict multiple paths with associated uncertainty.** Combining LSTMs with a Mixture Density Networks (MDN) as a final layer does not reduce overall performance and, in some cases, the accuracy was improved. The results shown that this approach performs well for predicting up to three paths per input tracklet. However, in some cases, when predicting four and five paths, some of the predicted paths begin to deviate further from the ground truth. This cannot be seen as a disadvantage since each predicted path has an associated uncertainty, so by looking at their probabilities the path with low probability can be discarded.

3. **Extensive exploration combining several contextual features in traffic scenarios to assess their impact in the path prediction task against only using x, y positional information.** Four different type of

context features were selected – object, ego-vehicle, social, and scene. These four type of features were selected since they cover most of the information that can be obtained from the perspective of an ego vehicle. The results shown that visual features of the objects and ego-vehicle features improved the performance against using only positional information.

4. **Exploration of different fusion strategies to evaluate their performance gain in the overall prediction task.** Three fusion strategies were evaluated – Early fusion of raw features, Early fusion of latent space features, and Middle fusion of latent space features. It was observed that both the information available and the fusion method are highly important. For each combination of features, using middle fusion on latent space features leads to better performance.

5. **Proposing an end-to-end architecture to represent and fuse holistic and contextual information normally present in traffic scenarios to predict the path of moving objects using LSTM and CNN architectures.** The proposed architecture consist in three main consideration. 1) To use latent space features to feed the used LSTMs. 2) To use middle fusion. 3) To use CNN models to extract visual features from images. The results evidenced that including those consideration in your model leads to error reduction in the path prediction task.

6. **Improvement of the performance of individual models in path predictions by building ensembles.** Three ensembles were built with the best models. Each ensemble uses different features, different models and the output of each model was combined in the ensemble by averaging in the final output layer. The results shown that the three ensembles improved the performance over the baseline methodology and the two other best individual models.

7.4 Future Work

Exploring the problem of predicting the future path of objects in traffic scenes led to interesting findings. However further work is needed in regard of these discoveries. This section outlines the aspects of this research thesis that can be extended as future work.

Use of more sophisticated architectures

- In the literature, more sophisticated architectures such as transformers or graph-based models can be found, this model can be explored to process the features used in this research.
- It was evidenced the impact that a CNN model has in the overall path prediction models. It would be interesting to evaluate more CNN models and see if some of them with common characteristic such as number of CNN layers, filter size, or specific type of layers (GAP, Attention, etc) perform better specifically in the context of path prediction task.
- In this research thesis only basic fusion strategies were explored. However, evaluation of more complex fusion strategies can be done. As a next step, it would be interesting to add a layer that automatically learns to select the important features fed to the model, instead of testing the combinations one by one, as it was done in this research work.

Explore more features that surround a vehicle in real life

- Representation of the interaction-aware features still pose a challenge. Explore better interaction-aware map features to encode the information of the other objects is needed.
- Encode static information of a scene, in this research the raw image of the scenes was used. There are now datasets were more information is given of a scene such as traffic lines, or raster images.

- Evaluate on bigger images to see if this improve the richness of the extracted features by CNNs models.
- Definitely, explore and understand the other existing datasets that provide information of trajectories of objects.

Intelligent filtering of multiple path prediction

An initial exploration of predicting multiple paths was shown in chapter 5. However, the approach used there only takes into account the previous observed tracklets to predict the multiple future paths. Future work is need to refine the predicted paths and discard those that are in regions no reachable by an object or type of object, i.e no object can be over a building, a tree or non reachable areas.

New evaluation metrics

As mentioned in 2.6 Evaluation metrics, the metrics used in this research thesis have certain limitations that are necessary to analyse and improve. Overcoming those limitation will allow us to create flexible models for path prediction. So it would be interesting to evaluate the metrics shown in [86] as future work.

7.5 Concluding Remarks

Road traffic collisions are an important cause of death and disability worldwide. Every year around the world 1.2 million people are killed and up to 50 million are injured or disabled as a result of road traffic collisions. According to the National Highway Traffic Safety Administration (NHTSA), 94% of road accidents are caused by human errors. Automobiles equipped with ADAS (Advanced Driver Assistance Systems) and sensors such as cameras, radars and LIDARs are now common place. Many of the accidents on the road can be avoided or at least can be mitigated by acting seconds in advance. For this reason, safety on the road is one of the main objectives in the development of ADAS. This thesis focused on developing a novel technique to accurately predict the future path of moving objects, such as pedestrians, vehicles, and cyclists based on data from egocentric cameras from a moving vehicle and incorporating contextual information normally found in traffic scenes.

In the author's opinion the most interesting findings are as follow. First, path prediction information can be treated as time series data, this way the prediction of the future locations of objects can be done by analysing the previous information of such objects. Second, LSTM architectures are suitable for path predictions due to their ability to leverage past information in a sequence to predict future values, i.e. observed past path to predict a future path. Third, LSTMs can be extended to predict multiple paths with associated uncertainty by combining them with Mixture Density Networks (MDNs) as a final layer. Fourth, it is very interesting that using different fusion strategies results in different results even when the same contextual information is being used. From here it could be seen that the two combination of features that improve the baseline model was 1. x, y, ego-vehicle features and 2. x, y, object image. In both cases, the fusion strategy that led to better performance was middle fusion on latent space representation. Finally, it was also highly interesting how using ensembles further improves the performance of the baseline model and the two best combination of features. The ensemble with the best results is the Ensemble 3 which includes the baseline methodology and the two best individual models.

The main constraints and challenges found in this research were the lack of annotated and synchronized object trajectory data. In addition, due to the use of sequential data with different contextual information limited hardware resources restricted the scale of experiments that could be completed. However, research in this important domain continues to advance with new techniques and datasets and will contribute to improved vehicle safety.

Bibliography

[1] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[2] Christopher M Bishop. "Mixture density networks". In: (1994).

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[4] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[5] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: (1999).

[6] Chiu-Feng Lin, A Galip Ulsoy, and David J LeBlanc. "Vehicle dynamics and external disturbance estimation for vehicle path prediction". In: *IEEE Transactions on Control Systems Technology* 8.3 (2000), pp. 508–518.

[7] Luke Fletcher, Lars Petersson, and Alexander Zelinsky. "Driver assistance systems based on vision in and out of vehicles". In: *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No. 03TH8683)*. IEEE. 2003, pp. 322–327.

[8] Dizan Vasquez and Thierry Fraichard. "Motion prediction for moving objects: a statistical approach". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. 2004, pp. 3931–3936.

[9] Paul Pilkington and Sanjay Kinra. "Effectiveness of speed cameras in preventing road traffic collisions and related casualties: systematic review". In: *Bmj* 330.7487 (2005), pp. 331–334.

[10] Paul D Yoo, Maria H Kim, and Tony Jan. "Machine learning techniques and use of event information for stock market prediction: A survey and evaluation". In: *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06).* Vol. 2. IEEE. 2005, pp. 835–841.

[11] Pifu Zhang et al. "Navigation with IMU/GPS/digital compass with unscented Kalman filter". In: *IEEE International Conference Mechatronics and Automation, 2005.* Vol. 3. IEEE. 2005, pp. 1497–1502.

[12] Raj Madhavan, Zeid Kootbally, and Craig Schlenoff. "Prediction in dynamic environments for autonomous on-road driving". In: *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on.* IEEE. 2006, pp. 1–6.

[13] James Colyar and John Halkias. "US highway 101 dataset". In: *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030* (2007).

[14] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. "Crowds by example". In: *Computer graphics forum.* Vol. 26. 3. Wiley Online Library. 2007, pp. 655–664.

[15] Jörg D Wichard and Maciej Ogorzałek. "Time series prediction with ensemble models applied to the CATS benchmark". In: *Neurocomputing* 70.13-15 (2007), pp. 2371–2378.

[16] Brendan T Morris and Mohan M Trivedi. "Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis". In: *Advanced Video and Signal Based Surveillance, 2008. AVSS'08. IEEE Fifth International Conference on.* IEEE. 2008, pp. 154–161.

[17] Chris Urmson et al. "Autonomous driving in urban environments: Boss and the urban challenge". In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.

[18] Andreas Ess et al. "Robust multiperson tracking from a mobile platform". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.10 (2009), pp. 1831–1846.

[19] Stefano Pellegrini et al. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 261–268.

[20] Nicholas I Sapankevych and Ravi Sankar. "Time series prediction using support vector machines: a survey". In: *IEEE Computational Intelligence Magazine* 4.2 (2009), pp. 24–38.

[21] World Health Organization. Dept. of Violence et al. *Global status report on road safety: time for action*. World Health Organization, 2009.

[22] Xin Li et al. "A multiple object tracking method using Kalman filter". In: *The 2010 IEEE international conference on information and automation*. IEEE. 2010, pp. 1862–1866.

[23] Piotr Dollar et al. "Pedestrian detection: An evaluation of the state of the art". In: *IEEE transactions on pattern analysis and machine intelligence* 34.4 (2011), pp. 743–761.

[24] Zhiqian Wu et al. "A path prediction method for human-accompanying mobile robot based on neural network". In: *International Conference on Intelligent Science and Intelligent Data Engineering*. Springer. 2011, pp. 35–42.

[25] Ramsey Faragher et al. "Understanding the basis of the Kalman filter via a simple and intuitive derivation". In: *IEEE Signal processing magazine* 29.5 (2012), pp. 128–132.

[26] Andreas Geiger et al. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[27] Alex Graves. “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850* (2013).

[28] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[29] Adam Houenou et al. “Vehicle trajectory prediction based on motion model and maneuver recognition”. In: *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2013, pp. 4363–4369.

[30] Christoph G Keller and Dariu M Gavrila. “Will the pedestrian cross? a study on pedestrian path prediction”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.2 (2013), pp. 494–506.

[31] Sebastian Koehler et al. “Stationary detection of the pedestrian? s intention at intersections”. In: *IEEE Intelligent Transportation Systems Magazine* 5.4 (2013), pp. 87–99.

[32] Hitesh A Patel and Darshak G Thakore. “Moving object tracking using kalman filter”. In: *International Journal of Computer Science and Mobile Computing* 2.4 (2013), pp. 326–332.

[33] Nicolas Schneider and Dariu M Gavrila. “Pedestrian path prediction with recursive Bayesian filters: A comparative study”. In: *German Conference on Pattern Recognition*. Springer. 2013, pp. 174–183.

[34] Sayanan Sivaraman and Mohan Manubhai Trivedi. “Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis”. In: *IEEE transactions on intelligent transportation systems* 14.4 (2013), pp. 1773–1795.

[35] BW Smith. “Summary of levels of driving automation for on-road vehicles”. In: *Center for Internet and Society, Stanford Law School* 1 (2013).

[36] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[37] Klaus Bengler et al. “Three decades of driver assistance systems: Review and future perspectives”. In: *IEEE Intelligent transportation systems magazine* 6.4 (2014), pp. 6–22.

[38] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[39] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[40] Christoph G Keller and Dariu M Gavrila. “Will the pedestrian cross? A study on pedestrian path prediction”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.2 (2014), pp. 494–506.

[41] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. “A survey on motion prediction and risk assessment for intelligent vehicles”. In: *ROBOMECH journal* 1.1 (2014), pp. 1–14.

[42] Xueheng Qiu et al. “Ensemble deep learning for regression and time series forecasting”. In: *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*. IEEE. 2014, pp. 1–6.

[43] Young-Woo Seo and Ragunathan Rajkumar. “Tracking and estimation of ego-vehicle’s state for lateral localization”. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2014, pp. 1251–1257.

[44] Ali Sharif Razavian et al. “CNN features off-the-shelf: an astounding baseline for recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813.

[45] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems.* 2014, pp. 3104–3112.

[46] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision.* Springer. 2014, pp. 818–833.

[47] Jan K Chorowski et al. “Attention-based models for speech recognition”. In: *Advances in neural information processing systems.* 2015, pp. 577–585.

[48] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning.* PMLR. 2015, pp. 448–456.

[49] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. “An empirical exploration of recurrent network architectures”. In: *International conference on machine learning.* 2015, pp. 2342–2350.

[50] Qiang Li et al. “Kalman filter and its application”. In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS).* IEEE. 2015, pp. 74–77.

[51] Zachary C Lipton, John Berkowitz, and Charles Elkan. “A critical review of recurrent neural networks for sequence learning”. In: *arXiv preprint arXiv:1506.00019* (2015).

[52] Andreas Møgelmose, Mohan M Trivedi, and Thomas B Moeslund. “Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations”. In: *2015 IEEE Intelligent Vehicles Symposium (IV).* IEEE. 2015, pp. 330–335.

[53] Hideki Shimada et al. “Implementation and evaluation of local dynamic map in safety driving systems”. In: *Journal of Transportation Technologies* 5.02 (2015), p. 102.

[54] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning.* 2015, pp. 2048–2057.

[55] Leon Yao and John Miller. “Tiny imagenet classification with convolutional neural networks”. In: *CS 231N* 2.5 (2015), p. 8.

[56] Yu Zheng. “Trajectory data mining: an overview”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 6.3 (2015), pp. 1–41.

[57] Alexandre Alahi et al. “Social LSTM: Human trajectory prediction in crowded spaces”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 961–971.

[58] Ahmad Ali et al. “Visual object tracking—classical and contemporary approaches”. In: *Frontiers of Computer Science* 10.1 (2016), pp. 167–188.

[59] B Allotta et al. “A new AUV navigation system exploiting unscented Kalman filter”. In: *Ocean Engineering* 113 (2016), pp. 121–132.

[60] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 3213–3223.

[61] Klaus Greff et al. “LSTM: A search space odyssey”. In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[62] Siyu Huang et al. “Deep learning driven visual path prediction from a single image”. In: *IEEE Transactions on Image Processing* 25.12 (2016), pp. 5892–5904.

[63] YoungJoon Yoo et al. “Visual path prediction in complex scenes with crowded moving objects”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 2668–2677.

[64] Mohamed Akram Zaytar and Chaker El Amrani. “Sequence to sequence weather forecasting with long short-term memory recurrent neural networks”. In: *International Journal of Computer Applications* 143.11 (2016), pp. 7–11.

[65] Bolei Zhou et al. “Learning deep features for discriminative localization”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929.

[66] Florent Altché and Arnaud de La Fortelle. “An LSTM network for highway trajectory prediction”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 353–359.

[67] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. “Long-term on-board prediction of pedestrians in traffic scenes”. In: *1st Conference on Robot Learning*. 2017.

[68] Rahul Dey and Fathi M Salemt. “Gate-variants of gated recurrent unit (GRU) neural networks”. In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[69] Arthur Emidio T Ferreira, Bruno Luiggi M Espinoza, and Flavio de Barros Vidal. “Predicting vehicle trajectories from surveillance video in a real scenario with Histogram of Oriented Gradient”. In: (2017).

[70] John Cristian Borges Gamboa. “Deep learning for time-series analysis”. In: *arXiv preprint arXiv:1701.01887* (2017).

[71] David Ha and Douglas Eck. “A neural representation of sketch drawings”. In: *arXiv preprint arXiv:1704.03477* (2017).

[72] Nikita Japuria, Golnaz Habibi, and Jonathan P How. “CASNSC: A context-based approach for accurate pedestrian motion prediction at intersections”. In: (2017).

[73] Naila Habib Khan and Awais Adnan. "Ego-motion estimation concepts, algorithms and challenges: an overview". In: *Multimedia Tools and Applications* 76.15 (2017), pp. 16581–16603.

[74] ByeoungDo Kim et al. "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 399–404.

[75] Shaoshan Liu et al. "Creating autonomous vehicle systems". In: *Synthesis Lectures on Computer Science* 6.1 (2017), pp. i–186.

[76] Apurva Narayan and Keith W Hipel. "Long short term memory networks for short-term electric load forecasting". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2017, pp. 2573–2578.

[77] Gerhard Neuhold et al. "The mapillary vistas dataset for semantic understanding of street scenes". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4990–4999.

[78] Derek J Phillips, Tim A Wheeler, and Mykel J Kochenderfer. "Generalizable intention prediction of human drivers at intersections". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1665–1670.

[79] Khaled Saleh, Mohammed Hossny, and Saeid Nahavandi. "Intent prediction of vulnerable road users from motion trajectories using stacked LSTM network". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 327–332.

[80] Atri Sarkar et al. "Trajectory prediction of traffic agents at urban intersections through learned interactions". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–8.

[81] Daksh Varshneya and G Srinivasaraghavan. “Human trajectory prediction using spatially aware deep attention models”. In: *arXiv preprint arXiv:1705.09436* (2017).

[82] Fred Wegman. “The future of road safety: A worldwide perspective”. In: *IATSS research* 40.2 (2017), pp. 66–71.

[83] Khaled A Althelaya, El-Sayed M El-Alfy, and Salahadin Mohammed. “Evaluation of bidirectional lstm for short-and long-term stock market prediction”. In: *2018 9th international conference on information and communication systems (ICICS)*. IEEE. 2018, pp. 151–156.

[84] Federico Bartoli et al. “Context-aware trajectory prediction”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 1941–1946.

[85] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. “Long-term on-board prediction of people in traffic scenes under uncertainty”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4194–4202.

[86] Jiang Bian et al. “A survey on trajectory clustering analysis”. In: *arXiv preprint arXiv:1802.06971* (2018).

[87] Nachiket Deo and Mohan M Trivedi. “Convolutional social pooling for vehicle trajectory prediction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1468–1476.

[88] Yaocong Hu, MingQi Lu, and Xiaobo Lu. “Spatial-Temporal Fusion Convolutional Neural Network for Simulated Driving Behavior Recognition”. In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE. 2018, pp. 1271–1277.

[89] Xinyu Huang et al. “The apolloscape dataset for autonomous driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 954–960.

[90] Ronny Hug et al. “Particle-based pedestrian path prediction using LSTM-MDL models”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2684–2691.

[91] Xue-Bo Jin et al. “State-of-the-Art Mobile Intelligence: Enabling Robots to Move Like Humans by Estimating Mobility with Artificial Intelligence”. In: *Applied Sciences* 8.3 (2018), p. 379.

[92] Antonios Karatzoglou, Adrian Jablonski, and Michael Beigl. “A Seq2Seq learning approach for modeling semantic trajectories and predicting the next location”. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2018, pp. 528–531.

[93] Salman Khan et al. “A guide to convolutional neural networks for computer vision”. In: *Synthesis Lectures on Computer Vision* 8.1 (2018), pp. 1–207.

[94] Seong Hyeon Park et al. “Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1672–1678.

[95] Inc. The MathWorks. *Deep Learning: Long Short-Term Memory Networks (LSTMs)*. `www.youtube.com/watch?v=5dMXyiWddYs`. Oct. 2018.

[96] Eric Thorn et al. *A framework for automated driving system testable cases and scenarios*. Tech. rep. United States. Department of Transportation. National Highway Traffic Safety . . ., 2018.

[97] Kaiping Xu et al. “Collision-free lstm for human trajectory prediction”. In: *International Conference on Multimedia Modeling*. Springer. 2018, pp. 106–116.

[98] Hao Xue, Du Q Huynh, and Mark Reynolds. “SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1186–1194.

[99] Takuma Yagi et al. "Future person localization in first-person videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7593–7602.

[100] Zoheb Abai and Nishad Rajmalwar. "DenseNet Models for Tiny ImageNet Classification". In: *arXiv preprint arXiv:1904.10429* (2019).

[101] Ariyan Bighashdel and Gijs Dubbelman. "A survey on path prediction techniques for vulnerable road users: From traditional to deep-learning approaches". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 1039–1046.

[102] Ming-Fang Chang et al. "Argoverse: 3d tracking and forecasting with rich maps". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8748–8757.

[103] Thomas Eiter et al. "Towards a semantically enriched local dynamic map". In: *International Journal of Intelligent Transportation Systems Research* 17.1 (2019), pp. 32–48.

[104] Kai Olav Ellefsen, Charles Patrick Martin, and Jim Torresen. "How do Mixture Density RNNs Predict the Future?" In: *arXiv preprint arXiv:1901.07859* (2019).

[105] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

[106] Lian Hou et al. "Interactive trajectory prediction of surrounding road users for autonomous driving using structural-LSTM network". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[107] Kai Huang et al. "A lightweight privacy-preserving CNN feature extraction framework for mobile sensing". In: *IEEE Transactions on Dependable and Secure Computing* (2019).

[108] R. Kesten et al. *Lyft Level 5 Perception Dataset 2020*. `https://level5.lyft.com/dataset/`. 2019.

[109] Yuexin Ma et al. "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 6120–6127.

[110] Ewoud AI Pool, Julian FP Kooij, and Dariu M Gavrila. "Context-based cyclist path prediction using recurrent neural networks". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 824–830.

[111] Alina Roitberg et al. "Analysis of deep fusion strategies for multi-modal gesture recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[112] Mohsin Sharif et al. "Tiny image classification using Four-Block convolutional neural network". In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2019, pp. 1–6.

[113] Zheng Tang et al. "CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[114] Alex Zyner, Stewart Worrall, and Eduardo Nebot. "Naturalistic driver intention and path prediction using recurrent neural networks". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[115] Holger Caesar et al. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11621–11631.

[116] Jing Gao et al. "A survey on deep learning for multimodal data fusion". In: *Neural Computation* 32.5 (2020), pp. 829–864.

[117] Christian Garbin, Xingquan Zhu, and Oge Marques. “Dropout vs. batch normalization: an empirical study of their impact to deep learning”. In: *Multimedia Tools and Applications* (2020), pp. 1–39.

[118] John Houston et al. “One Thousand and One Hours: Self-driving Motion Prediction Dataset”. In: *arXiv preprint arXiv:2006.14480* (2020).

[119] Alex Sherstinsky. “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network”. In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[120] Pei Sun et al. “Scalability in perception for autonomous driving: Waymo open dataset”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 2446–2454.

[121] Neo Wu et al. “Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case”. In: *arXiv preprint arXiv:2001.08317* (2020).

[122] Fisher Yu et al. “BDD100K: A diverse driving dataset for heterogeneous multitask learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 2636–2645.

[123] Ekim Yurtsever et al. “A survey of autonomous driving: Common practices and emerging technologies”. In: *IEEE Access* 8 (2020), pp. 58443–58469.

[124] MIT. *MIT 6.S191 Introduction to Deep Learning.* URL: `http://introtodeeplearning.com/`. (accessed: 20.02.2021).

[125] Christopher Olah. *Understanding LSTM Networks.* URL: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. (accessed: 01.09.2019).

[126] Anna Shcherbina. “Tiny ImageNet Challenge”. In: ().

[127] Hao Xue. *SS-LSTM model for pedestrian trajectory prediction.* URL: `https://github.com/xuehaouwa/SS-LSTM/`. (accessed: 20.09.2020).