UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Farhan Syakir

Multihead Attention Enhanced Memory Augmented Neural Network for Multimodal Trajectory Prediction

Master's Thesis (30 ECTS)

Supervisor(s):
Naveed Muhammad, PhD
Yar Muhammad, PhD

Tartu 2021

Multihead Attention Enhanced Memory Augmented Neural Network for Multimodal Trajectory Prediction

Abstract:

Autonomous driving has gathered an increased interest over the last two decades. One of the problems in autonomous driving that the researchers are actively trying to solve is agent trajectory prediction. The trajectory prediction is the problem of predicting future trajectories of surrounding agents such as other cars, cyclists, pedestrians, and any other road users around an autonomous vehicle. Deep learning has shown promising results in tackling the problem. There are various deep learning approaches addressed to the problem, and one of the approaches is using Memory Augmented Neural Network (MANN) and multi-head attention layer. Memory augmented neural networks in multimodal trajectory prediction have been proposed in the literature to address trajectory prediction (in a model called memory augmented networks for multiple trajectory prediction or MANTRA), but they do not use multi-head attention layers. Meanwhile, multi-head attention layers have also been investigated in the literature but in different contexts within this research topic.

In this work we proposed two models which both employ multi-head attention layers to the memory augmented neural network model. We name the models Multihead Attention Enhanced MANN (MAEMANN) 1 and MAEMANN-2. Similar to MANTRA, MAEMANN uses AutoEncoder, Memory Controller, and iterative refinement module (IRM). While the AutoEncoder and Memory Controller is responsible for memory, the IRM compiles the output from the memory and input from the surrounding agents in the environment. The MAEMANN-1 uses the multi-head self-attention layer in the memory network to improve predicting future trajectory by giving attention to the multiple neighboring memories, while MAEMANN-2 uses the multi-head attention in IRM to improve perceiving surrounding agents. Our experimental results showed that both MAEMANNs (i.e. models 1 and 2) outperform the MANTRA model, when tested on the Kitti dataset, where we predict 4 seconds future trajectory given 2 seconds past. In the multimodal prediction where the number of modes is 5, the MAEMANN-1 improves the Final Displacement Error (FDE) and Average Displacement Error (ADE) at $t = 4$ *seconds* by 10.58 % and 9.24%. Meanwhile, for MAEMANN-2, the improvements for FDE and ADE are 14.39% and 13.47%.

Keywords:
Trajectory prediction, Memory Augmented Neural Network, Transformer, Autonomous Driving

CERCS:
P170 (Computer science, numerical analysis, systems, control)

Mitmepealine täiustatud tähelepanu mäluga närvivõrk multimodaalse trajektoori ennustamiseks

Lühikokkuvõte:

Autonoomse juhtimise tehnoloogia on viimase kahe aastakümnega palju populaarsust kogunud. Üks probleem autonoomse juhtimise tehnoloogia juures, mida teadlased püüavad lahendada, on agentide trajektooride ennustamine. Trajektoori ennustamise probleemi all mõeldakse ennustamist, milline on agentide nagu autode, jalgratturite, jalakäijate ja teiste sõidutee kasutajate, tuleviku trajektoor. Sügavõppe kasutamisega on saadud selle probleemi lahendamisel edukaid tulemusi. Trajektoori ennustamiseks on olemas mitu varianti sügavõppest ning üks neist kasutab mälu rikastamisega närvivõrku (*Memory Augmented Neural Network,* MANN*)* ja mitmepealist tähelepanukihti (*multi-head attention layer*). Kirjanduses on soovitatud multimodaalse trajektoori ennustamiseks kasutada mälu augmenteerimisega närvivõrku (nt mudel nimega *memory augmented networks for multiple trajectory prediction,* MANTRA), aga see ei kasuta mitmepealist tähelepanukihti. Mitmepealisi tähelepanukihte on samuti kirjanduses uuritud, aga see valdkond ei kattu antud uurimistöö teemaga.

Selles töös on kirjeldatud kahte mudelit, mis kasutavad mõlemad mitmepealist tähelepanu kihti ja mälu rikastamisega närvivärku. Antud tööd nimetatakse neid *Multihead Attention Enhanced* MANN (MAEMANN) 1 ja MAEMANN 2. Sarnaselt MANTRA-le, kasutab MAEMANN autoenkoodrit, mälukontrollerit ja iteratiivset täiustusmoodulit (*iterative refinement module,* IRM). Autoenkoode ja mälukontroller vastutavad mälu kasutamise eest ning IRM kompileerib mälust saadava väljundi ja sisendi, mis saadakse ümbritsevas keskkonnas olevatelt agentidelt. MAEMANN 1 kasutab mitmepealist tähelepanukihti mäluvõrgus, et täiendada tuleviku trajektoori ennustust, pöörates tähelepanu mitmele naabruses olevale mälule. MAEMANN 2 aga kasutab mitmepealist tähelepanukihti IRM-is, et täiustada ümbruskonnas olevate agentide tuvastamist. Läbi viidud eksperimentide tulemused näitavad, et mõlemad MAEMANN mudelid 1 ja 2 edestavad MANTRA mudelit, kui neid kasutada Kitti andmestiku peal ja ennustades 4 sekundipikkust tuleviku trajektoori, kui on teada 2 sekundipikkused andmed minevikust. Multimodaalse ennustuse puhul, kus ennustatavaid variante on 5, on MAEMANN 1 tulemus parem 10.58% ja 9.24% võrra, kui meetrikaks kasutada vastavalt lõplikku mõõteviga (*Final Displacement Error,* FDE) ja keskmist mõõteviga (*Average Displacement Error*, ADE) kohal $t = 4$ *sekundit*. MAEMANN 2 tulemused on aga FDE ja ADE puhul paremad vastavalt 14.39% ja 13.47%.

Võtmesõnad:
Trajektoori ennustamine, mälu laiendatud närvivõrk, transformaator, autonoomne sõit

CERCS:
P170 (arvutiteadus, numbriline analüüs, süsteemid, juhtimine)

Table of Contents

1. Introduction

Improving safety and reducing the number of accidents on the road are important issues. Progressing on those topics could protect many lives, for example, the driver, the passenger, the pedestrian, even the animals on the road. While there are a lot of efforts on that, the main cause of the accident on the road is human error, often caused by the driver. According to The National Motor Vehicle Crash Causation Survey (NMVCCS), based on the sample they have, 94% of crashes from 2005 to 2007 were because of drivers' errors [1]. Therefore the idea of replacing the driver with the autonomous driving system is promising.

Another reason to replace the driver with an autonomous system is to increase accessibility and efficiency. A lot of people can not drive vehicles themself, for example, children, old people, or people with disabilities. With the help of autonomous systems, they do not need a human driver, thus making their life easier.

There are various approaches regarding the architecture of the autonomous driving system. The end-to-end driving system, where the model receives direct sensory input from camera and sensors and outputs directly to actuators such as the steering wheel and gas pedal [2]. Such a model processes the entire pipeline as a black box for the machine to solve and does not have an explicit trajectory prediction part.

Meanwhile, the other approach that has great potential is to make a trajectory prediction as one part of the autonomous system. A trajectory prediction model expects another model to handle the perception part. The sensors and cameras input are fed to a perception model, and the result passes on to the trajectory prediction model. Therefore, a trajectory prediction model is only responsible for predicting trajectories of agents surrounding the autonomous vehicle. Not only that, there are also numbers of available datasets that are manually annotated that can be used for this approach, for example, Kitti [3], Argoverse [4], and Lyft [5] datasets. Thus, based on its remarkable potential and the dataset's availability, it was decided to investigate such an approach in this thesis project.

A lot of literature exists on trajectory prediction models. A paper that reviews conventional physics-based models taking up on the trajectory prediction problem has been published as [6]. The methods they survey are Hidden Markov Models, Support Vector Machines, and Dynamic Bayesian Networks. However, another survey paper shows that Deep learning models presented superior performance compared to the physics-based model [7].

The most common deep learning layer in trajectory prediction models is Recurrent Neural Network (RNN) based and Convolutional Neural Network (CNN) based layer [7]. The CNN performs well for perceiving grid-like data such as an image, and it is common to represent the input for the trajectory prediction as an image. RNN is good at perceiving sequential data or timestamp data, which is how the input in the trajectory prediction is represented. For RNN based layers, commonly Long Short Term Memory (LSTM) [8] or Gated Recurrent Unit (GRU) [9] are used.

However, there are novel types of deep learning layers that have great potential for trajectory prediction research such as Transformer [10], Graph Neural Network (GNN) [11], and Memory Augmented Neural Network (MANN) [12]. One of the networks that have a lot of potentials, but has not been investigated to that extent in the literature is MANN. MANN for trajectory prediction has been presented in [13]. The potential comes from how it can mimic how a driver behaves in new cases based on previous cases they had in their memory. They query the closest trajectory from the memory using the new past trajectory as a key to make it

an anchor to predict the new future trajectory. Therefore, the research carried out in this thesis focuses on such networks.

The attention layer has also gained recognition in machine learning [10]. The attention layer gives a model the ability to give more consideration to the more important part of the input, and it is also part of transformers. The attention mechanism is gaining success in NLP topics, such as language translation [10] and question answering [14]. In trajectory prediction topics, the attention layer has also been shown to perform well [15], [16].

Based on the potential of the MANN and the promising results of the attention mechanism, we decided to adopt those technologies in this thesis. Moreover, to the best of our knowledge, there is no published research that investigates combined MANN and attention mechanisms for trajectory prediction problems. Therefore, in this thesis, we propose Multihead Attention Enhanced MANN (MAEMANN) 1 and MAEMANN-2, an enhanced MANN with the multi-head attention layer, for the purpose of trajectory prediction in autonomous driving.

The main contributions of this thesis are:

- It proposes two MAEMANN models, a modification of the existing memory augmented networks for multiple trajectory prediction (MANTRA) [13] model using the multi-head attention mechanism.
    - The first model (that we refer to as model-1 in the thesis) is the multihead-attention enhancement on the decoder on the memory network.
    - The second MAEMANN model (that we refer to as model-2) is the multihead-attention enhancement on the Iterative Refinement Module (IRM).
- It presents hyperparameter experiments with the MAEMANN model presenting the best combination of the parameters for the two MAEMANN models.
- We show the comparison of the performance of MAEMANN-1 and MAEMANN-2 with MANTRA as well as other baseline models that include Kalman filter, simple linear layer network, and multi-layer perceptron (MLP) .

The structure of the remainder of the thesis is as follows. For Chapter 2 we explained the terms and notations. The background is discussed in chapter 3. Chapter 4 explains the methodology. Experiment results and future works are presented and examined in chapter 5. Finally, chapter 6 concludes the thesis.

2. Terms and Notations

To avoid confusion on explaining the concept of this thesis, we define some terms as such:

- Target Vehicle (TV): is the vehicle whose behavior we are trying to predict.
- Surrounding Vehicle (SV): are the vehicles whose behavior is investigated by the prediction model since it may have an impact on how TV behaves in the future.
- Agents: generic entities surrounding the TV and possibly affecting the TV behavior. Agents could be other vehicles, pedestrians, or cyclists.
- Self Attention: Throughout this thesis, the term attention or self-attention is always referring to multi-head self-attention. Since we are not using single-head attention and always use self-object as input to the attention.
- Unimodal: The model predicts only one trajectory.
- Multimodal: The model predicts five different trajectories *(k = 5)* and evaluates only the best trajectory.
- *k*: Notation *k* is the number of prediction modes generated by a multimodal model and mentioned in this thesis multiple times. If the *k* is 1 then the model is a unimodal model.

3. Background

The background chapter consists of six parts. The trajectory prediction problem and its challenges are explained in section 3.1. After that, we present the input representation categorization in trajectory prediction. We explore the map representation categorization in section 3.3. In section 3.4 we discuss the categorization of output representation in trajectory prediction. Evaluation metrics are presented in section 3.5. Section 3.6 discusses how deep learning in general advances the trajectory prediction problem.

3.1. Trajectory Prediction

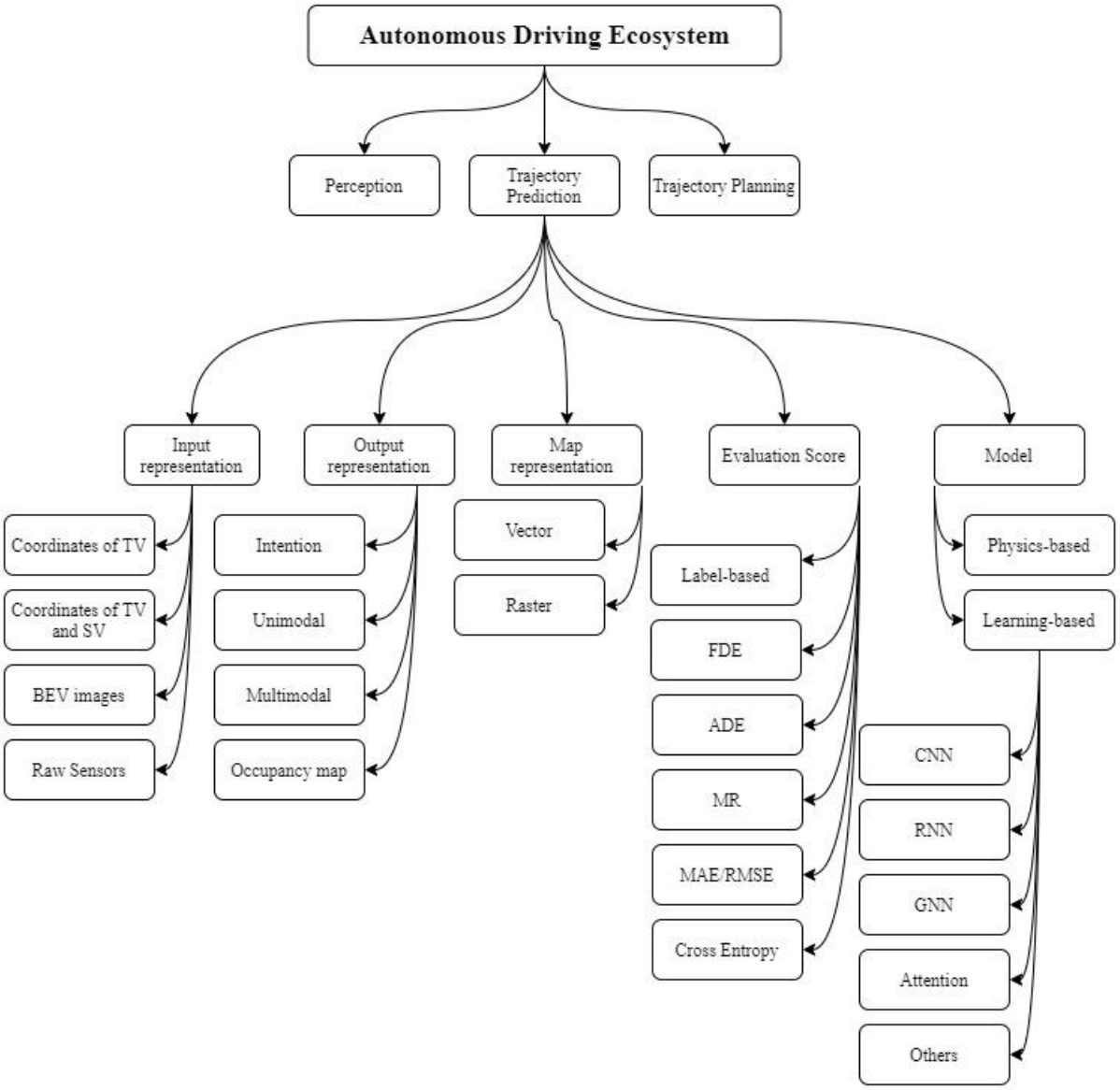

Figure 1. Trajectory Prediction on Autonomous Driving Ecosystem

Figure 1 is the overview of the trajectory prediction problem in the autonomous driving ecosystem with its categorization on multiple aspects. The image above is inspired by multiple sources such as [2]–[4], [11]. Trajectory prediction's position in the autonomous driving ecosystem is in the middle, after the perception module, and before trajectory planning.

Trajectory prediction's goal is to predict the future trajectory of an object given its past trajectories and the surrounding agent states. The input for trajectory prediction models is processed images or Spatio-temporal data of the environment from the perception module or from the annotated dataset. Inputs from raw sensors or raw images from cameras are processed by the perception module to detect and localize objects in the environment. The trajectory prediction output is used as an input for the autonomous driving system's trajectory planning module to make further decisions. While trajectory prediction is also applicable to other agents such as pedestrians [17], this thesis focuses on the trajectory prediction of vehicles.

There are many challenges involved in trajectory prediction for vehicles. The physical limitations of the vehicles need to be considered, for example, a maximum angle that a vehicle can afford when it is turning, especially considering the size of the vehicle. A big vehicle will need a bigger angle and vice versa. Another challenge is that vehicles have to follow certain rules on the road, as well as interact with surrounding agents gracefully. Another challenge is that a trajectory prediction model is also expected to be multimodal i.e. it is able to give multiple possible future trajectories for other agents [7], in order for the autonomous driving system to get richer input for the decisions and planning module. Moreover, since the data in this context is spatio-temporal data with a complex structure, developing the trajectory prediction model is complicated and consumes a lot of computational resources.

Figure 1 presents a summary of different aspects involved in a trajectory prediction module, within an autonomous driving system. We divide the categorization in trajectory prediction problems into five groups: the input representations, the output representations, the map representation, the evaluation score, and the modeling approach. Those groups are discussed in detail in the following sections.

3.2. Input Representation

In terms of an input representation, we follow the categorization presented by [7] because based on sources we explored, all of them fall into categories presented in their paper. In the trajectory prediction problem, input representations are generally of four types, which are track history of TV, track history of TV and SVs, Bird's Eye View (BEV), and raw representation.

However, we slightly differ from the categorization provided by [7]. While in the paper the history of TV and SV disregards its surrounding environment and solely depends on coordinates, in practice we find that including the environment map as an input in training is inevitable. Moreover, a method like a refinement module can include the environment map as an additional input. Therefore this TV history category covers TV history inputs and TV history inputs with maps. However, we decided not to create more categories such as TV history with maps because it seems unnecessary.

For the input on track history, some research for example [8], [18] predicts the future trajectory using only intrinsic data of the TV (e.g. position, velocity, angle, etc). While those papers do not consider the impact of the environment, other papers such as [13], [19] use both the history of TV and its environment as multiple input from the dataset, for example in [13] the track history input goes to its RNN and map input goes to its CNN.

The next category, or type, is the input from the history of TV and SVs. In this category, SVs can be used by the model as anchors to predict the TV's future trajectory. However,

researchers differ on the best number i.e. on how many SVs are needed for trajectory reference. Some papers use six SVs [20], [21], others use seven [22] and nine SVs [23]. However, those earlier research works do not consider the environment as input and depend only on features from TV and SVs, meanwhile recent research [24]–[26] uses the environment.

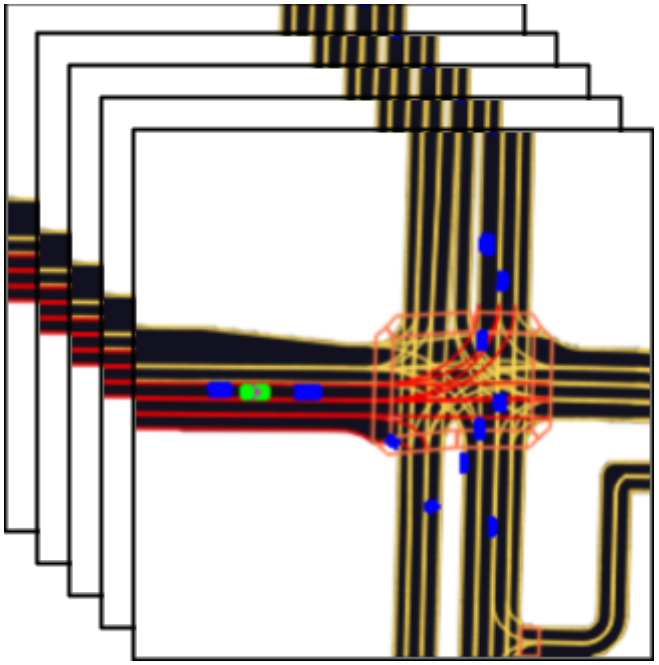

Figure 2. BEV input consists of n * H * W input [26]

Another category for input representation is BEV. In this category, the input of the model is not the position of the TV or SV, but the sequence of images, that are probably in batches that portray the condition of the traffic using color-coded shapes as depicted in Figure 2. This BEV input comes from the perception model's output and it is dependent on the performance of the perception model. An erroneous perception model could potentially lead to erroneous trajectory prediction. However, the BEV is able to capture the interaction between agents and is flexible in portraying the environment. Some works that use BEV are presented in [5], [27], [28]

The last category is using the raw sensory data as an input, whether it is data from sensors or from cameras as videos [29]. It has some advantages such as not depending on the previous model, and ensuring there is no information loss. For this approach, a larger computational resource is needed to process raw input data.

To wrap the section, this thesis uses the first category of input representation. Since we use the history of TV and map representation where the history of TV goes to the autoencoder and the map goes to the refinement module. There might be an SVs object in the map but we do not use its history, since that data is not available in the dataset (more information about the dataset is presented later in the thesis, in section 5.2).

3.3. Map Representation

While in earlier research map representation is in BEV input representation, in more recent research the model can have different input pipelines, one for history and the other for the map, which makes every category in input representation put a map on it. We decided it

makes sense to separate the map representation as one category to group trajectory prediction models. Thus, we divide the map representation into two categories: a Rasterized and a Vector-based map.

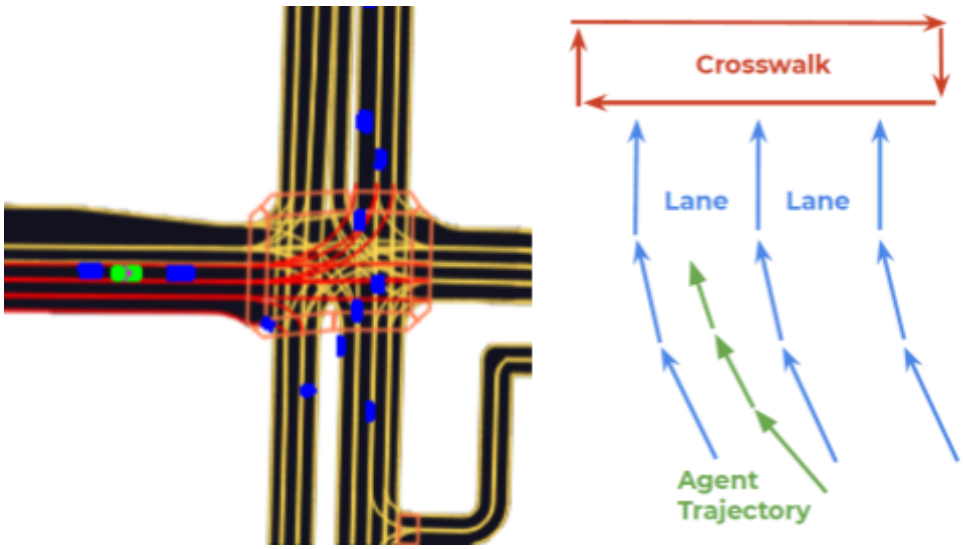

Figure 3. Rasterized map (left) [26] vs Vector-based Map (right) [27]

Rasterized map, as depicted in Figure 4, is the most common representation in the literature. It is basically a pixelated image and it is used in earlier trajectory prediction models [5], [27], [28]. This representation is simple but it costs more space and computational power compared to other map representations.

In more recent papers, the polyline map representation is gaining popularity. Since it costs less space and only has fewer features to compute than the raster method that uses every pixel as a feature. Some papers combine the polylines map with GNN [30], [31], while some others combine it with attention-based models like [24], [32].

In this thesis, we follow the rasterized map. While it is possible to use polyline maps, extensive implementation needs to be done to achieve that, it was out of scope for the current thesis due to time constraints.

3.4. Output

Most of the literature we surveyed divides the output representations in a consistent manner. The output of autonomous trajectory prediction can be grouped into four categories: intention, unimodal trajectory, multimodal trajectory, and occupancy map.

The first category is the intention-based output. The intention is the most simple output, where the output is a label such as “left”, “right”, “stop” [33], or even “west”, “east”, “south”, etc [8]. This output gives a high level of understanding that could be provided to the user and also has a low computational cost. However, this approach is not sufficient for the next model input which is the trajectory planning model. Moreover, the label of the output is usually pre-defined and not flexible.

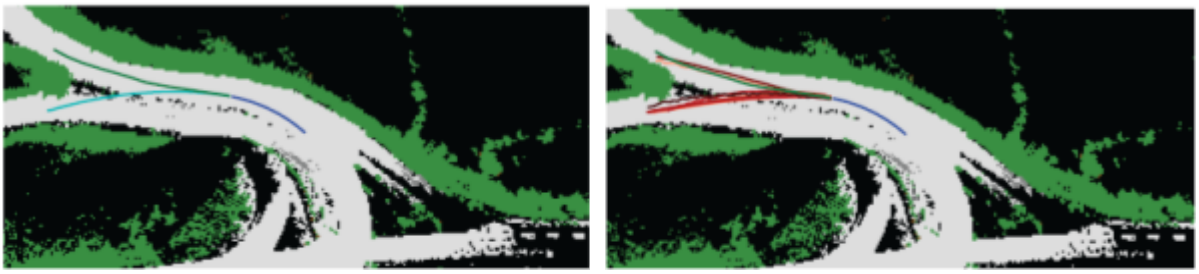

Figure 4. Unimodal output (left) vs multimodal output (right) [13]

From Figure 4, we can see the left image is the unimodal output. In the image, the blue line is the past trajectory and the dark green line is the ground truth and the light blue is the prediction. As it can be seen, the model only outputs one prediction compared to the right image where the model outputs multiple predictions. The unimodal output provides more information than the intention category as it gives the trajectory prediction. However, the unimodal output is still not sufficient for future trajectory prediction as it fails to predict two equally probable trajectories.

The next output type is the multimodal trajectories. As we can see in Figure 4 on the right image, while the unimodal prediction only provides one trajectory the multimodal prediction provides multiple trajectories. The advantage of this output representation is it might fully represent the vehicle behavior. Between works in the literature, there is a difference in the implementation of multimodal output. Some papers use a static number of modes [13], [21]. Other papers use a dynamic number of modes [19]. Some papers provide probability on each mode [34], some do not [13], [19].

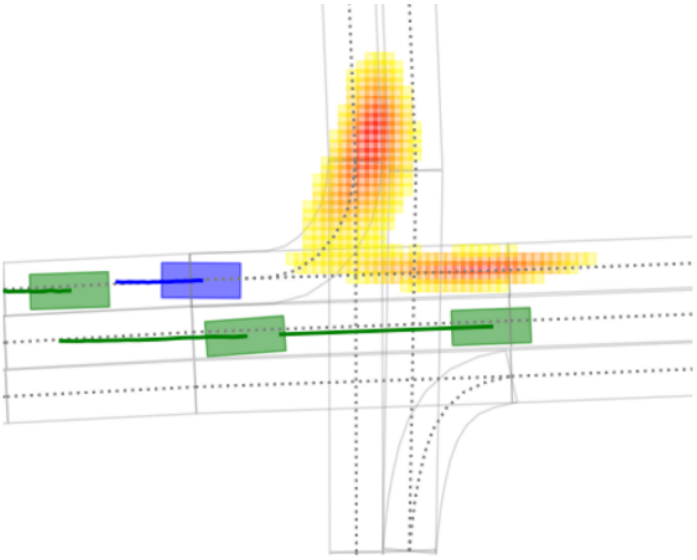

Figure 5. Grid occupancy map output [21]

Another promising output type, or category, is the occupancy map. In this approach instead of predicting trajectory, the model outputs the probability of occupancy on each grid cell in the map [27]. The illustration of grid occupancy is represented in the heatmap in Figure 5. Same as the previous category, this category has the potential to represent all trajectory predictions and provide rich information. Also, the output is flexible, for example in [26], the occupancy map is converted to multimodal trajectory predictions.

Finally, to wrap this section, the work presented in this thesis uses unimodal and multimodal trajectory prediction output. We decided to use these output types to be able to compare this thesis with other models that also use the same output.

3.5. Evaluation Metrics

The choice of evaluation metrics is usually linked to the output representation. Since the most common output type of trajectory prediction models is unimodal and multimodal trajectories, therefore in this subsection focus on evaluation metrics around that output type.

However, it is useful to note that if the output of the trajectory is an intention label such as "left", "right", "stop", etc. then evaluation metrics are related to label prediction metrics, such as accuracy, precision, recall, f1-score, Negative Log-Likelihood (NLL), etc.

For unimodal and multimodal trajectory prediction, other metrics such as Miss Rate (MR), Average Displacement Error (ADE), Final Displacement Error (FDE), cross-entropy, or NLL, Mean Absolute Error (MAE), or Root Mean Squared Error (RMSE) are more suitable.

For MR, a threshold is usually defined, and then when a predicted trajectory falls outside the threshold it is counted as missed. The number of "miss" trajectories compared to all trajectories is the miss rate. Another case is when the output is provided in the form of a probability. In this case the MR is known as p-MR where the difference is the miss times with (1.0 - p) as the contribution where p corresponds to the probability of the best-forecasted trajectory.

The next evaluation metrics are ADE and FDE. This thesis uses these evaluations. Displacement error is the distance between the predicted future and the ground truth. ADE is the average displacement error from timestamp 0 to n and FDE is the displacement error at a given timestamp (it will be explained in more detail in section 5.1). However, if the probability for future prediction is given then the metrics are added $min(-\ log(p),\ -\ log(0.05))$ to the given distance where the p is the probability error. In the multimodal case, the model will pick the minimum ADE or FDE out of the multimodal prediction the model gives. The issue with these evaluation metrics is they incentivize multimodal models without probability to predict more guesses as there is no penalty for giving too many guesses.

Evaluation metrics for trajectory prediction that are also common in regression problems are MAE and RMSE. Both work similarly, and MAE is similar to ADE. The error that needs to be measured is the distance error between prediction and ground truth. Another evaluation metric that is commonly used for regression problems and has been used in the Lyft competition is cross-entropy or NLL [35].

3.6. Trajectory Prediction Model

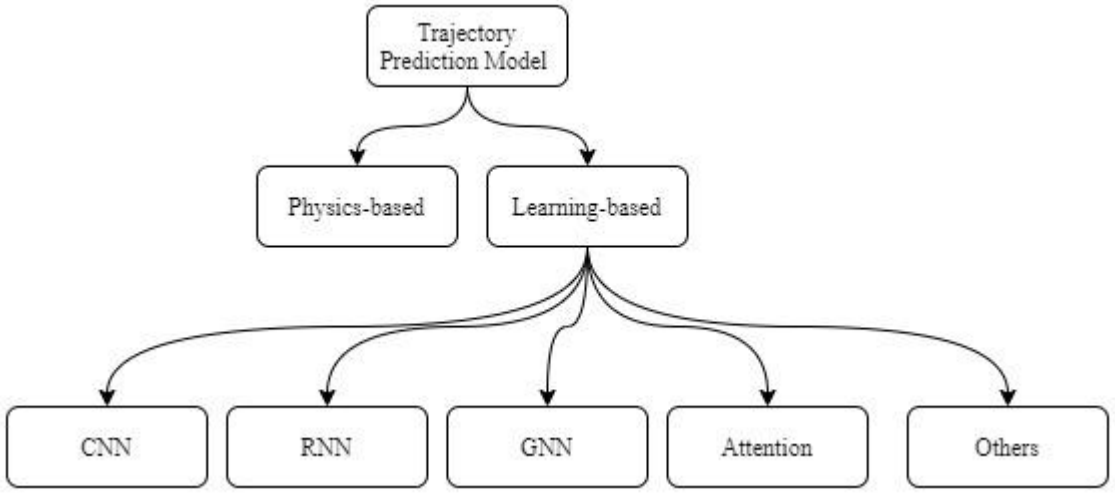

Figure 6. Trajectory prediction model categorization

As presented in Figure 6, we divide trajectory prediction models into two categories. The first one is the physics-based model and the second is the learning-based model. The first category is the model that is handcrafted from existing physical models and the other is machine learning or deep learning model where the machine is given a supervised dataset to learn to be able to predict future trajectories.

Physics-Based Trajectory Prediction Model

The physics-based model is the simplest kind of trajectory model prediction, which assumes that vehicles move according to the laws of physics. In this category, the model is manually hand-crafted according to existing rules. Examples of prediction models in this category are the constant velocity, the constant acceleration, and the Kalman Filter.

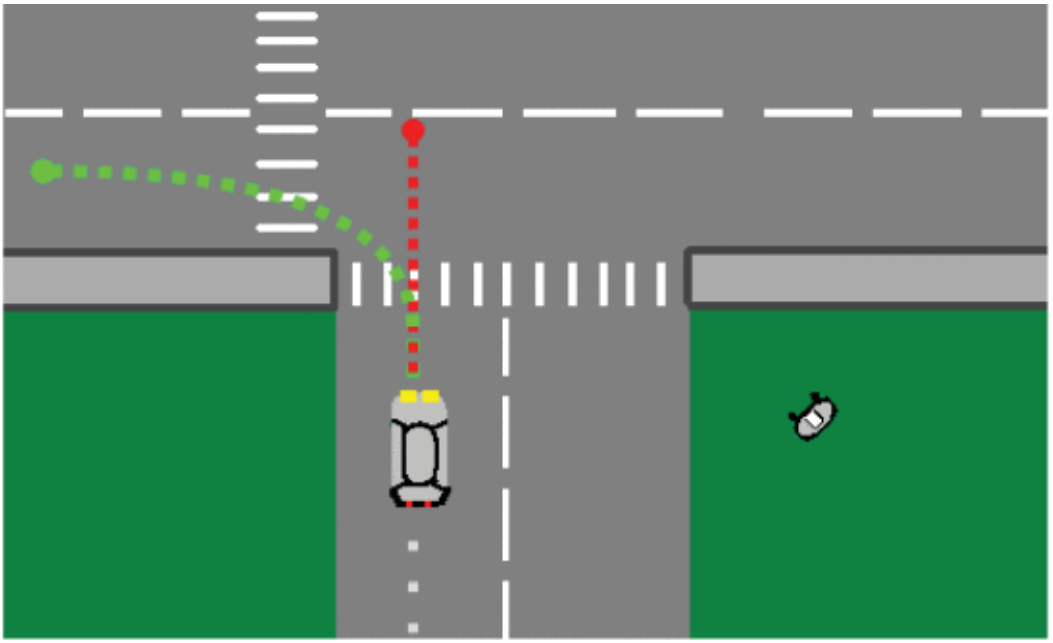

Figure 7. Physic based trajectory prediction using constant velocity, green is the ground truth and red is prediction [36]

From Figure 7, which represents the constant velocity model, we can see that despite the shape of the road, the model predicts the vehicle to constantly go straight to the middle of the road because of the current state (velocity, angle, etc) implied so. This model does not consider the intention of the driver, let alone the other agents' intention on the roads.

As mentioned earlier the limitation of a physics-based model is they assume no change of action considering current circumstances of the road. This is because such models work solely on low-level properties, so it is typical for them to not be able to anticipate maneuvers such as stopping, turning, or slowing down. While such methods have been significantly used in early research, they are not favored in recent years, and the learning-based models are gaining favor because of more promising performance.

Learning-Based Models in Trajectory Prediction

Machine learning models perform significantly better compared to manually crafted models because of their nature to capture real-life details that could be missed by a rule-based algorithm [6]. Subtle interactions between many agents in one timeframe are impractical to be crafted by engineers. In terms of agility to change, a machine learning approach is better. While a rule-based algorithm, given a new bigger dataset, would require manual changes and analysis to add more features to it or to handle more corner cases, a machine learning model would need to make a small adjustment so the model can handle the new data with retraining.

To the best of our knowledge, there is no comparative study between deep learning and other machine learning models in trajectory prediction problems. Deep learning is performing better than machine learning in other areas of research such as medical [37], [38], and natural language processing (NLP) [39], [40]. Because of that, we decided to favor deep learning instead of conventional machine learning models because of its great potential and performance in other application areas.

As in Figure 6, we divide learning-based trajectory prediction models into five categories. Each of these can be a stand-alone prediction model or just a layer inside a larger model. In recent years, it is common in the literature that a model combines different layers and consists of more than one type of layer in this categorization. The five categories are Convolution Neural Network (CNN), Recurrent Neural Network (RNN), Graph Neural Network (GNN), Attention Network, and other networks. We discuss this in more detail as follows.

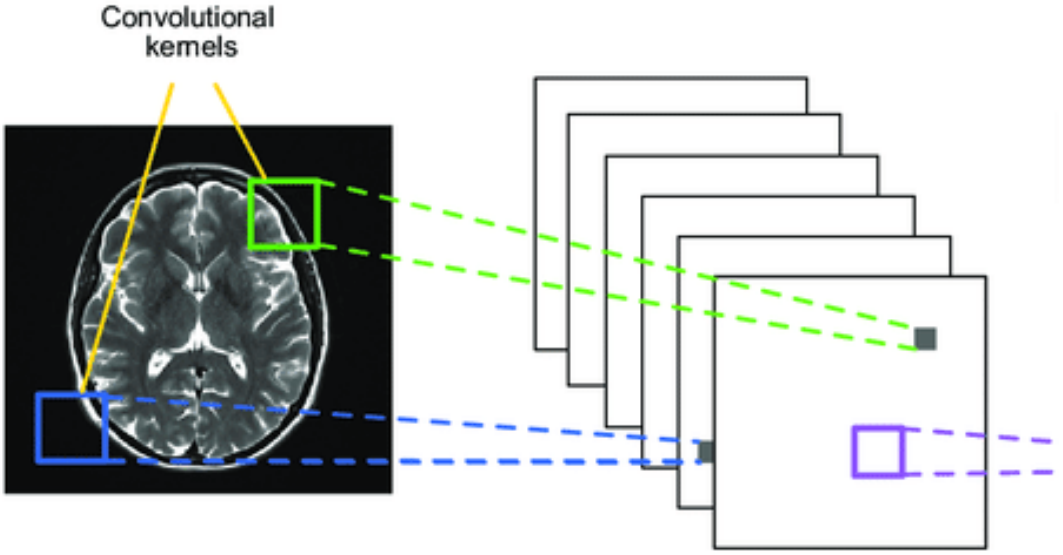

Figure 8. CNN Illustration [41]

The first group is CNN. The CNN is a model that consists of one or more convolving layers. As it can be seen from Figure 8, convolving layers will extract features from a window of an input image (it is also possible for more than 2-dimensional input) and pass it as an output for the next layer. CNN is very common for processing image input as the input of trajectory prediction mostly has images. Moreover, if the input category is BEV. Therefore it is common to utilize CNN for trajectory prediction [8], [34], [42].

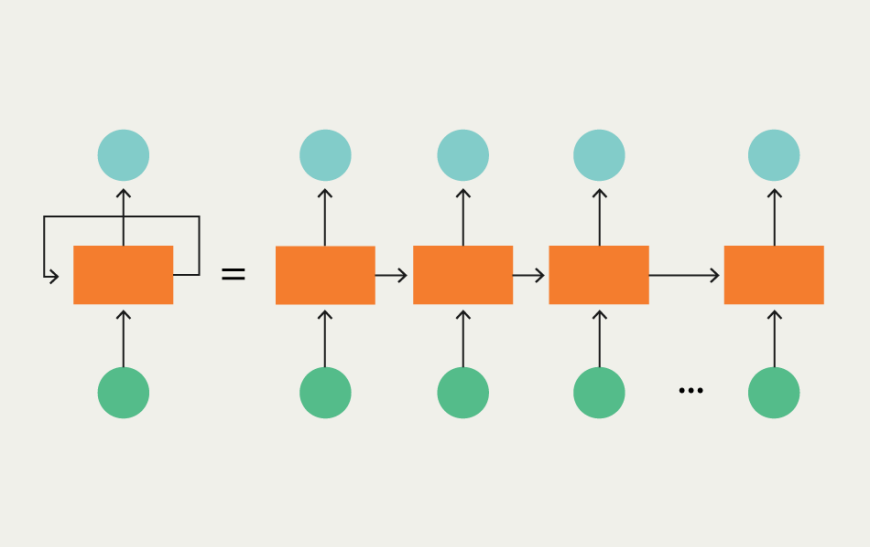

Figure 9. RNN illustration [43]

Another model category that is commonly used in trajectory prediction is RNN-based models. As illustrated in Figure 9, RNN is a model that specializes in recognizing a sequential pattern as it takes input from the previous sequence. Therefore it is suitable for RNN to handle input such as the history of TV and SV, and it can even be combined with CNN to handle the map input. In trajectory prediction, there are multiple options to pick, because there are different RNN-based layer types such as the original RNN [18], LSTM [8], [21], [25], and GRU [13], [44].

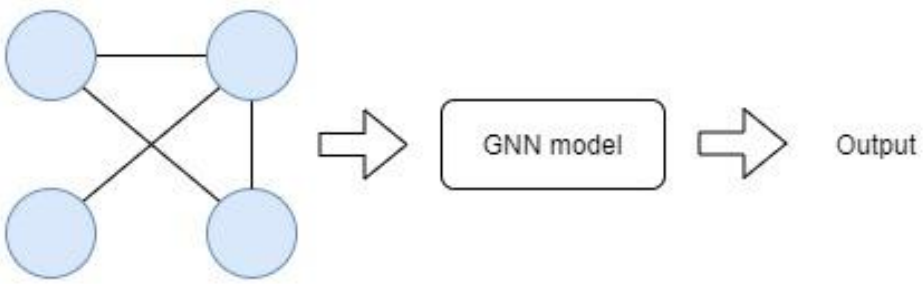

Figure 10. GNN Illustration

Another category that has gained popularity in recent years is the GNN. The GNN model is a machine learning model designed to receive a graph as an input and then produce an output as depicted in Figure 10. This approach is suitable to represent a real traffic condition as the road can be depicted as nodes and edges, and interaction between agents on the road can be represented by nodes and edges. Some models that use the GNN concept are Graph Convolutional Network (GCN) [45], Graph Attention Network (GAN) [46], and VectorNet [30].

The next two model categories are attention networks and memory networks. Since this thesis uses both networks as the main focus then we will discuss it in detail in the next subsection.

Memory Network in Trajectory Prediction

Memory usage in deep learning networks has existed for a while, represented by RNN based networks for example LSTM and GRU. In those models, however, memory is represented by a single hidden state vector that stores all temporal information. As a result, memory can only be addressed as a whole, and they lack the ability to address individual knowledge elements, which is required for algorithmic manipulation and quick inference. Furthermore, these models are unreliable in simulating long-term prediction due to their mechanism that updates the state on each timestep.

Memory augmented neural networks (MANN) were first proposed in [12]. A MANN is a type of neural network with an external memory component that can be read and written using controllers. Because it is external and not modified at every timestamp this memory can be designed, accessed, and analyzed as much as needed. The MANN also has the ability to query from the memory using a key to make it not tied to a predefined hidden size. As a result, giving them the flexibility to store, remove or manipulate memory according to what it might need.

Multiple research works about memory networks have been published recently. Memory network has been demonstrated to be working well with NLP tasks. MANN has been shown in [12], [47] to be capable of successfully addressing Question Answering challenges, in which the model must answer questions given a series of sentences.

In this thesis, we based our work around MANN. The reason we decided to explore MANN is that MANN is unique and provides a lot of potential benefits, such as flexibility in configuring and implementing memory management, interpretability on the memory, and mimicking the behavior of the memory of the driver. However, not much research on trajectory prediction using this model has been published to the best of our knowledge, despite very few works that we describe as follows.

A research work that uses MANN in trajectory prediction is memory augmented networks for multiple trajectory prediction (MANTRA) [13]. MANTRA uses a key-value approach to store the past and future representations. Then later, query the future as an anchor point in the testing part where the future ground truth is not available. In this thesis, we take the MANTRA implementation and enhance it using an attention mechanism since the attention mechanism has a lot of benefits (presented in the next section) and it is gaining popularity among trajectory prediction models.

Attention Mechanism in Trajectory Prediction

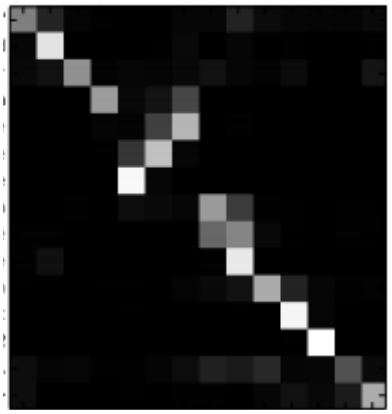

Figure 11. Heat map [48]

Attention mechanism was first introduced in [18] and became state-of-the-art in Natural Language Processing (NLP). The attention mechanism utilizes math formulas that transform a feature matrix into an attention matrix which is a matrix that has weights of attention between features. Figure 11 shows a heat map of one dimension of size n feature that becomes a two-dimensional attention matrix of size $n * n$ (that consists of the weight of attention between features). The black color means no attention while the brighter color means more attention. This attention concept is an exceptional success in various NLP tasks, like machine translation [18]–[20], image caption generation [21], text generation [22], and question answering [19].

Moreover, transformers also show promising results in computer vision tasks. For example, in image detection, transformers outperform the convolution layer [51]. On the other hand, other researchers combine transformers with convolution networks to produce a robust model [52] for image detection. Other computer vision tasks for which the transformer has been employed include tracking [53], classification [52], [54], [55], and image captioning [58]. While computer vision is more related to the perception part in an autonomous driving system, transformers also have the potential to be employed for trajectory planning.

Recent works have used variants of self-attention and transformers for modeling various designs in behavior modeling. Attention techniques such as trajectory encoding and decoding have been used by several researchers [15], [16], [56]. Others [16], [17], [57] employ it to encode relationships between agents and the scene. Other approaches such as [24] use a transformer to forecast the vehicle's permissible trajectory. The state-of-the-art trajectory prediction that uses a combination of networks also puts multi-head attention mechanisms as the central part [17]. Therefore in this thesis, we investigate the idea of using a multihead self-attention layer to the MANTRA model.

4. Methodology

This chapter presents our proposed methodology for trajectory prediction and consists of three sections. The first section explains the problem formulation. The global architecture is discussed in the second section and it is worth mentioning that we have two enhancement models that we will discuss in detail there. After that, we examine the components of the model. The hyperparameter configuration is discussed in the last section.

4.1. Problem Formulation

We decided to use five as the number of modes as used in the literature such as MANTRA [13]. Higher modes of $k = 10$ and $k = 20$ have also been used in the literature, but we chose to limit our method to fine because in many cases k is kept small, for example, $k = 1$ and $k = 3$ in Lyft trajectory competition [35], and in Argoverse $k = 1$ and $k = 6$ were used [49].

The goal of this thesis is to estimate the future trajectory given the past trajectory and the environment's context. We formulate our problem as such:

$$P(\tilde{x}^{f} \mid x^{p}, c) \tag{1}$$

$$\widehat{x^{f}} = \{\tilde{x}_{t}^{f}, \tilde{x}_{t+1}^{f} \ldots, \tilde{x}_{t+m}^{f}\}^{N} \tag{2}$$

$$x^{p} = \{x_{0}^{p}, x_{1}^{p} \ldots, x_{t}^{p}\} \tag{3}$$

Equation. (1) represents the main problem of the topic where we have to estimate the multimodal future trajectory $\widehat{x^{f}}$ given the past trajectory x^{p} and context c. The multimodal future trajectory prediction is presented in Eq. (2) where the x represents the position of the vehicle, the t is the current timestamp, the m is the future prediction window timestamp, and the N is the number of multimodal predictions. We formulate the past trajectory x^{p} in Eq. (3) where it consists of positions of the vehicle from timestamp 0 to t.

4.2. Global Architecture

In this section, we explained two enhancement MANTRA models using Multi-Head Attention (MHA). The first model is adding self-attention to the memory network thus giving the model attention mechanism to the neighboring memory. Meanwhile, the second model is adding multi-head attention to the iterative refinement module (IRM), thus giving the model attention mechanism to other agents surrounding the environment. In addition to the two models that we present in the thesis, we also investigated several other modifications using a multi-head attention layer, but they are out of the scope of this document. The two models covered in the thesis showed result improvement in both unimodal and multimodal cases.

While the following section will explain the model in general, the model consists of many components. The explanation about the components will be discussed in detail in section 4.3.

Multihead Attention on Memory

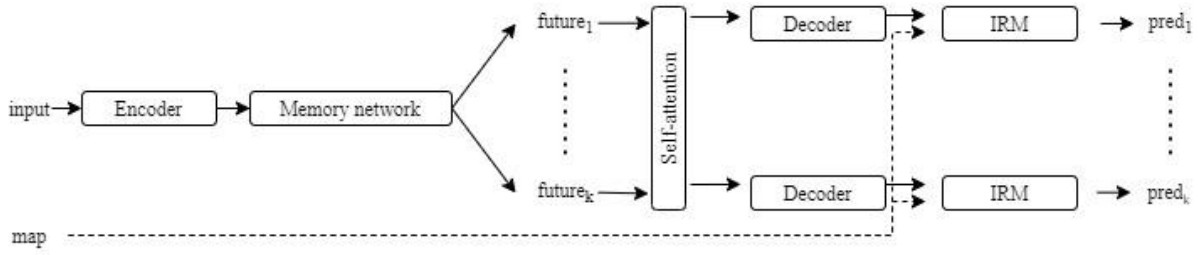

Figure 12. The architecture of model-1 (MHA enhancement on memory network)

The first model is MHA enhancement on the memory network depicted in Figure 12. As it can be noted from the figure, the input goes to the encoder and then feeds the encoded input to the memory network. The memory network then will query to the memory using the encoded input to retrieve the encoded future. In the multimodal setup, we will query k numbers of encoded future trajectories. Then those future trajectories are merged into one tensor which is then passed to the self-attention layer. After that, the tensor goes to the decoder. Then the output tensor from the decoder goes along with map input to the IRM (iterative refinement module) and produces the future trajectory predictions.

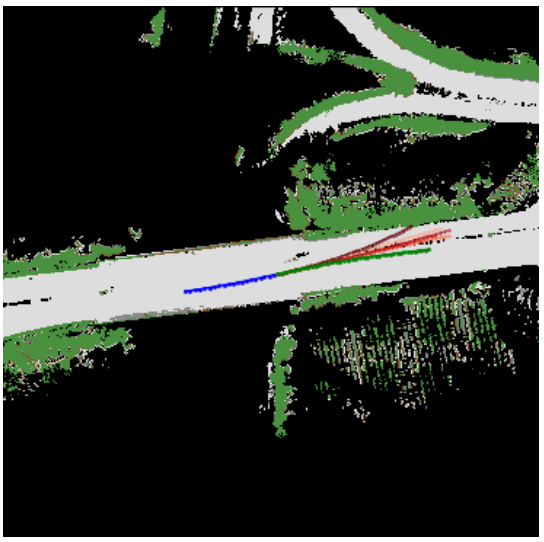

Figure 13. An example of multiple trajectory prediction, blue is the past trajectory, green is the ground truth, other colors are the predictions for $k = 5$.

The enhancement comes from self-attention layer addition since the original MANTRA model does not have that. We used self-attention so the model will be able to peek at the neighboring queried encoded future trajectories. Similarly to how a real driver might remember a couple of scenes in their memory and use it as an anchor. Then, if there are a lot of similar actions the driver took in the past from multiple scenes, it is probably what the driver will take in the future. For example in Figure 13, given the past trajectory (blue), the TV might go to a couple of different trajectories, but most of them just go straight, and by giving the model information about the other encoded predictions the model might give a better prediction.

Multihead Attention on IRM

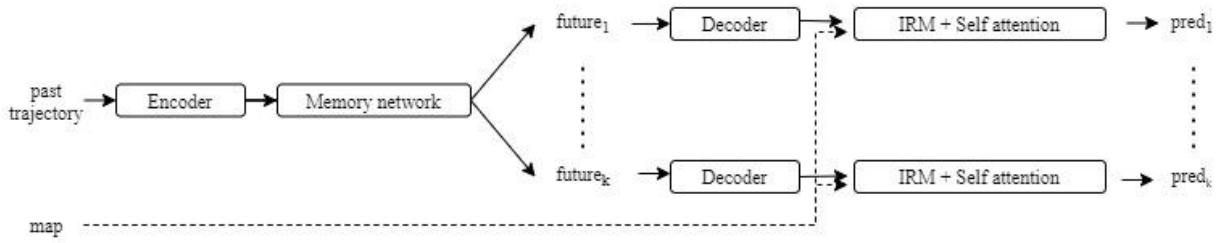

Figure 14. The architecture of model-2 (MHA enhancement on IRM)

The second model is MHA enhancement on the IRM as depicted in Figure 14. For this model, the difference is the position of the self-attention layer. In this model, the self-attention layer is inside the IRM. The IRM uses an encoded future and a map as inputs. Also in the module, the model processes the interaction between the TV and the environment. By adding an attention mechanism to the model, the model might have a better understanding of the surroundings and thus give better trajectory prediction.

4.3. Model Components

This section discussed the components of the model. Since previously, we discussed the general architecture of two models that share a lot of the same components, below we explain the components in more detail. The first component is the memory network. After that, the encoder is explained in the second section. In the third section, we present the self-attention layer. In the fourth section, we discuss the memory controller module. The decoder is discussed in the fifth section. The last section covers the IRM.

Memory Network

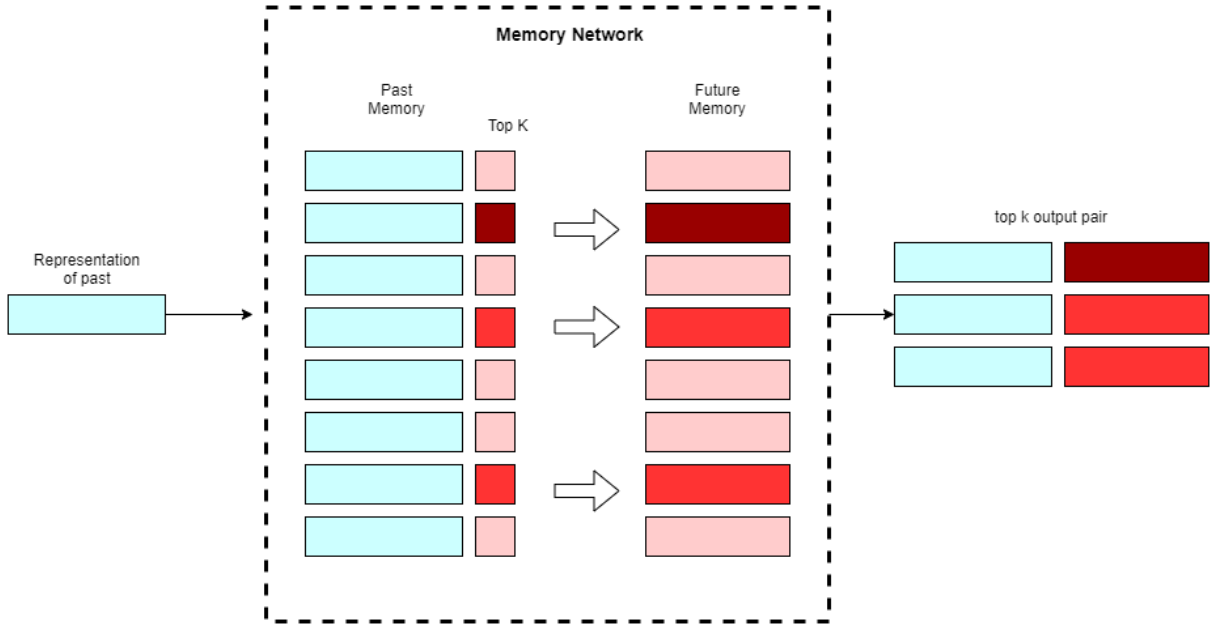

Figure 15. Memory Network

This is the main layer of the model, and we want to mimic the driver behavior on querying the memory. As shown in Figure 15, we use the key-value-based memory, where the key is the encoded representation of the past, and the value is the encoded representation for the future. Let $[x_p^i, x_f^i]$, where x_p^i and x_f^i are the past and the future trajectories based on timestamps. Then, we define π as the past encoder function where $\pi^i = \Pi(x_p^i)$. Meanwhile, φ is the future trajectory encoder function where $\varphi^i = \Phi(x_f^i)$.

Key-value memory is an encoded version of past and future trajectories defined as $M = \{\pi^i, \varphi^i\}$, thus $|M|$ is the number pairs in the memory. Therefore given past trajectory x_p^k, we encode it to the π^k then use it to retrieve the samples from memory. The past encoded trajectory π^k is used in the lookup (addressing) stage, whereas the future trajectory φ^k is used in the reading (returning the result) stage. In the addressing part, we decide to use the cosine similarity function, which output s_i where:

$$s_i = \frac{\pi^k \pi^i}{\|\pi^k\| \|\pi^i\|} \tag{4}$$

After getting the s_i across the memory, we pick the top-*k* future trajectories as an anchor and then feed them into the next layer. Note that the number of *k* is the number of the future predictions we want to retrieve. This method enables the model to forecast the correct future by looking forward to possible options. Also, the encoding process gives a more effective saving mechanism rather than saving it raw. Lastly, the future decoder is notated as $\psi^i = \Psi(\pi^i, \varphi^j)$, because the future decoder receives the pair of encoded past and the future and outputs the predicted future trajectories.

AutoEncoder

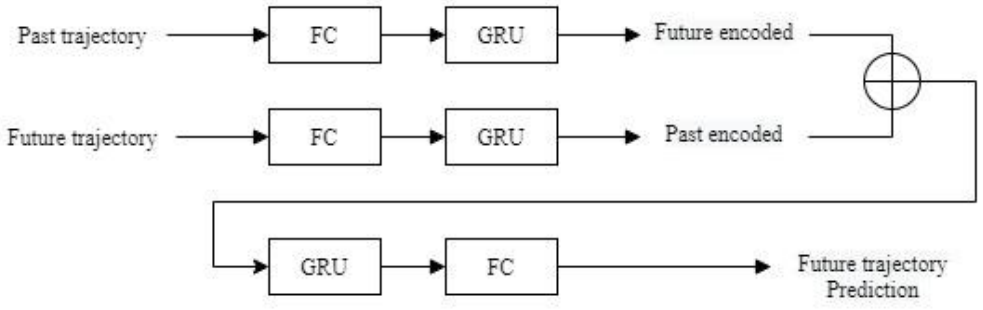

Figure 16. AutoEncoder structure

The autoencoder consists of two parts, the encoder, and the decoder. The training process consists of three stages of training, which will be discussed in detail in section 4.4. It is important to mention that in the first stage of training the encoder and decoder will be pre-trained as an autoencoder as the position depicted in Figure 16, while in the third stage, the encoder and decoder will be positioned separately as in Figure 12 and Figure 14.

Table 1. Parameters of autoencoder

Model	Layer	Past		Future		Parameters
		Input	Output	Input	Output	
Encoder	FC	x_p^i	$(x_p^i)'$	x_f^i	$(x_f^i)'$	Input size= 20 (past), 40 (future) Output size= 32
	GRU	$(x_p^i)'$	π^i	$(x_f^i)'$	φ^i	Input size= 32 Hidden size= 48
Concat						
Layer	**Model**	**Input**		**Output**		**Parameters**
Decoder	FC	(π^i, φ^i)		$(\pi^i, \varphi^i)'$		Input size= 96 Hidden size= 96
	GRU	$(\pi^i, \varphi^i)'$		x_f^i		Input size= 96 Output size= 2

In the pre-training phase, the autoencoder will convert the past trajectory x_p^i and future trajectory x_f^i to a past trajectory encoding π^i and a future trajectory encoding φ^i. The autoencoder model structure is shown in Figure 16 and its parameter in Table 1. The past and the future trajectory moves to a Fully Connected (FC) network. The FC network has an input size of 20 and an output size of 32. Note that both go to different FC networks. After that, they go to the GRU layer with a hidden size of 48. The output from the past's GRU becomes the representation of the past. The output of the future's GRU becomes the representation of the future. After that, we concatenate both representations then we feed them to decoder GRU with the hidden size of 96. The output from the decoder goes to the FC network as the final trajectory prediction.

As mentioned previously, the encoding-decoding functions Π, Φ, Ψ are learned as an autoencoder together. The encoders learn to convert past and future points into meaningful representations, while the decoders learn to duplicate the future. The reconstruction process is conditioned with an encoding of the past rather than solely using the future as input. The autoencoder network gets both past and future as inputs then produces a future trajectory. The autoencoder serves as a converter for the past and future. After the pretraining, we will have fully trained Π, Φ, Ψ functions for the next training phase.

Self Attention

The self-attention layer allows the vehicle to give attention to important features of the input. This method is explained in [10] in the context of NLP, where it is used to translate sentences. The formulation of the attention mechanism is as follows:

$$\text{Output} = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{5}$$

The input matrices for self-attention are *Q, K,* and *V* where Q is the query matrix, K is the key matrix, and V is the value matrix. From Eq. 5, first, we calculate the dot product of *Q* and *K* to get the selection of keys, then scale the result using the square root of the d_k. After that, the result gets normalized using the softmax function. Finally, we calculate the result with *V* to get the attention matrix. By using the same Q, K, and V as an input that we refer to as self, now we have a self-attention mechanism. The process that we talked about here is the single-head self-attention mechanism.

Furthermore, in the same paper i.e. [10], the authors propose a way to improve the single-head self-attention model which is called multi-head self-attention. The first step is slicing the input of the attention, and forming multiple heads of attention. Then, the multiple heads of attention run together parallelly. Finally, results are averaged to get the final output.

Table 2. Self-attention for model-1

Layer	Input	Output	Parameter
Self-attention	φ^{i}	$\varphi^{i,}$	$d_{model} = n_{modes} * n_{hidden} * 2$ $n_{head} = hyperparam$ $n_{layers} = hyperparam$

As we mentioned in the global architecture section, we have two different models where both of them have different ways of putting the attention layer on them. The first model is depicted in Figure 12, where the input is the encoded future and the output is the encoded future with attention weight, where it goes to the IRM. The parameter of the self-attention model in model-1 is presented in Table 2. The size of d_{model} is the size of modes times the size of hidden dimension times two because we have two dimensions input x and y. For the size of the head and the layers, we will try different combinations as hyperparameters and present the result in chapter 5.

Table 3. Self-attention for model-2

Layer	Input	Output	Parameter
Self-attention	φ^{i}	$\varphi^{i,}$	$d_{model} = 16$ $n_{head} = hyperparam$ $n_{layers} = hyperparam$

Meanwhile, the second model is presented in Figure 14 where the self-attention is inside the IRM. The parameter of model-2 is shown in Table 3. The size of d_{model} is 16 because the previous layer inside the IRM layer output's size is 16. The inside of the IRM will be discussed later in this chapter. For the size of the head and the layers, similar to model-1, we will experiment with them as hyperparameters and present the results of the experiment in chapter 5.

Memory Controller

In this thesis, we use the memory controller from the MANTRA [13]. The goal of the memory controller is to regulate a newly discovered memory. Given a brand-new case, it has to decide whether the case is different enough from any existing cases in the memory. Also, different from the question-answering problem in NLP, where the memory is wiped out for every case, a memory network in trajectory prediction has to be able to store cases permanently, thus growing the memory size over time. Therefore, the memory controller also acts as a filter for redundant memories thus decreasing the size of the memory to an optimal size.

The memory controller's goal is to output the $P(w)$probability of writing to the memory given a future trajectory and existing memory. Since the memory controller depends not only on the input sample, but also on the current state of the memory, and the state of the memory is growing over time, we can not feed the memory state directly to the memory controller. Therefore we use the pretrained autoencoder and memory network to pick the most similar existing trajectory using the cosine similarity function in Eq. 4. After that, we use the reconstruction error function as follows:

$$e = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathrm{r}_i(\hat{x}_f, x_f) \tag{6}$$

$$L_c = e\cdot(1 - P(w)) + (1 - e)\cdot P(w) \tag{7}$$

Where the e (Eq. 6) is the reconstruction error. N is the size of the future trajectory which is 40. The $\hat{x}_f$ and x_f both are prediction future trajectory and ground truth future trajectory. The $\mathrm{r}_i(\hat{x}_f, x_f)$ is the threshold function on the i-th point where the function equals 1 if the distance is within the threshold and 0 otherwise. While the threshold is linearly increasing from 0 at 0 seconds and 2 meters at 4 seconds. Furthermore, we want the memory controller to give the lesser $P(w)$ given larger e and vice versa, thus we formulate the loss controller function as $L_c$in Eq. 7.

Table 4. Memory Controller Parameter

Layer	Input	Output	Parameter
FC layer + Sigmoid	e	$P(w)$	$n_{input} = 1$ $n_{output} = 1$

The architecture of the memory controller consists of a fully connected layer and sigmoid and the parameter is depicted in Table 4. Given the input of reconstruction error, the model should output the writing probability. The model has an input size of 1 and an output size of 1 as well.

This memory controller will be trained in the second pre-training process because the process needs a pre-trained encoder and decoder to train the memory controller. The pretrained memory controller then will be used as one of the modules in the final training. A detailed explanation of the second pre-training will be discussed in section 4.4.

Iterative Refinement Module (IRM)

Similar to MANTRA [13] and DESIRE [19], we use IRM to refine future trajectory prediction. The IRM uses an iterative procedure. The goal of the IRM is to capture interaction features of agent-to-agent interaction and agent-to-environment sculpture. Given a map scene and future trajectory prediction the IRM produces the final output of refined future trajectory prediction.

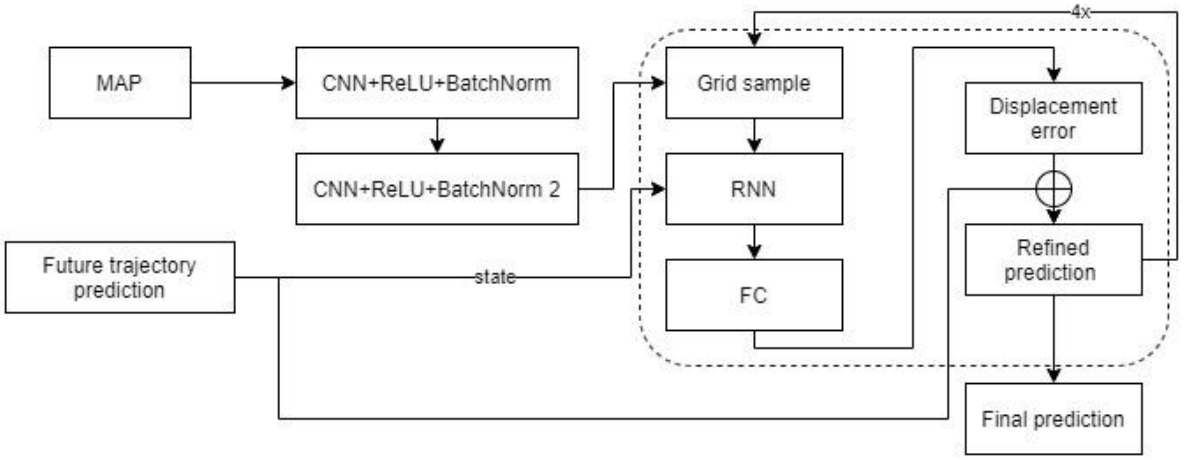

Figure 17. Structure of The IRM's model-1

Table 5. Parameter of IRM

Layer	Parameter
CNN+ReLU+BatchNorm	$n_{input} = 4; n_{output} = 8; n_{kernel} = 5; n_{stride} = 2;$ $n_{padding} = 2$
CNN+ReLU+BatchNorm 2	$n_{input} = 8; n_{output} = 16; n_{kernel} = 5; n_{stride} = 1;$ $n_{padding} = 2$
Grid sample	-
RNN	$n_{input} = 16; n_{output} = 96$
FC	$n_{input} = 96; n_{output} = n_{future} \cdot 2$

In this thesis, we have two different IRMs, for model-1 and model-2. The structure of the IRM in model-1 is depicted in Figure 17 and the parameters in Table 5. The IRM in model-1 is considered identical to the IRM in MANTRA. Given an input of a map scene and future trajectory prediction, the model should output the final prediction. The map goes to two CNN layers both wrapped with ReLU and BatchNorm. After that, the output goes to the iterative module in the dotted square. The output from the CNN layer goes to the grid sampling module to extract its interaction feature. Then the RNN gets input from the grid sampling

module and predicts future trajectory. After that, the output is the displacement error that is added to future trajectory prediction as a new future trajectory prediction. The process inside the dotted box was repeated four times to get a refined final future trajectory prediction.

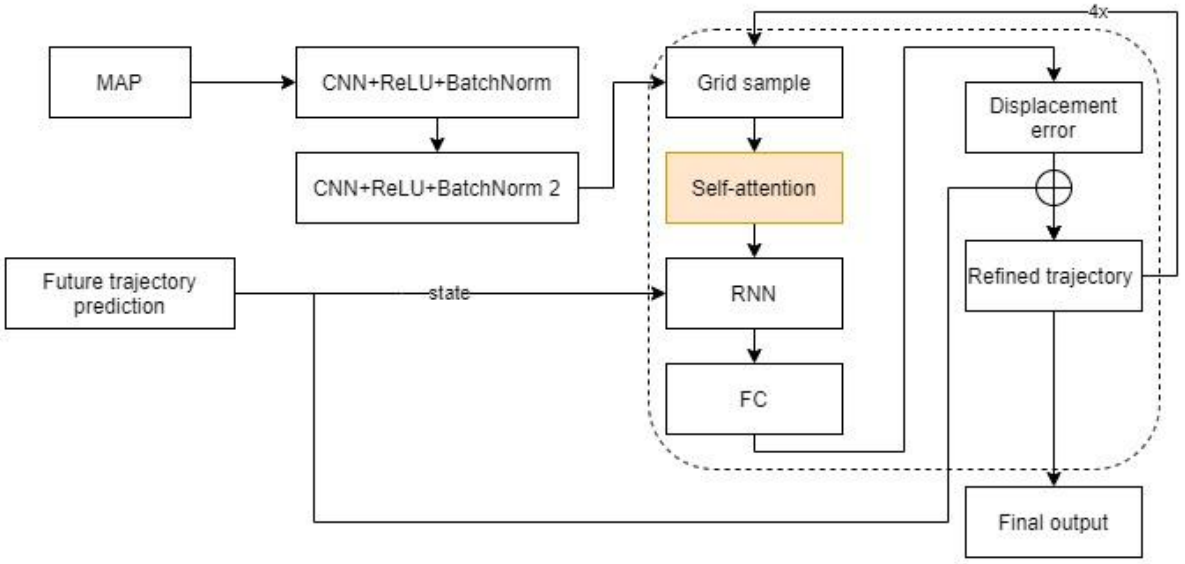

Figure 18. Structure of IRM's model-2

The Structure of The IRM's model-2 is shown in Figure 18. The implementation of the IRM in model-2 is identical to the IRM in model-1, except for the self-attention layer before RNN. The parameters of the IRM are the same as model-1 and presented in Table 5, except for the additional self-attention parameters that are presented in Table 3. Also, from Figure 18 and Table 5, we can infer the reason for the hardcoded size of d_{model} of 16, which is because of the output size from the CNN-2.

4.4. Training

The training consists of 3 phases, pretraining 1, pretraining 2, and final training. After which the testing is carried out. Given 2 seconds of past trajectory, we predict the 4 seconds of future trajectory. For the default training parameters, we used a maximum epoch of 600, the error function of Mean Squared Error (MSE), and adam optimizer with a learning rate of 10^{-5}.

Pretraining 1 is straightforward, we trained the autoencoder to predict future trajectory given the past and the future trajectories. It is worth mentioning that we used ground truth as an input to predict ground truth. The goal of this training is to create a decent past encoder, future encoder, and future decoder which can encode and decode the trajectory into the matrix that suits the memory network.

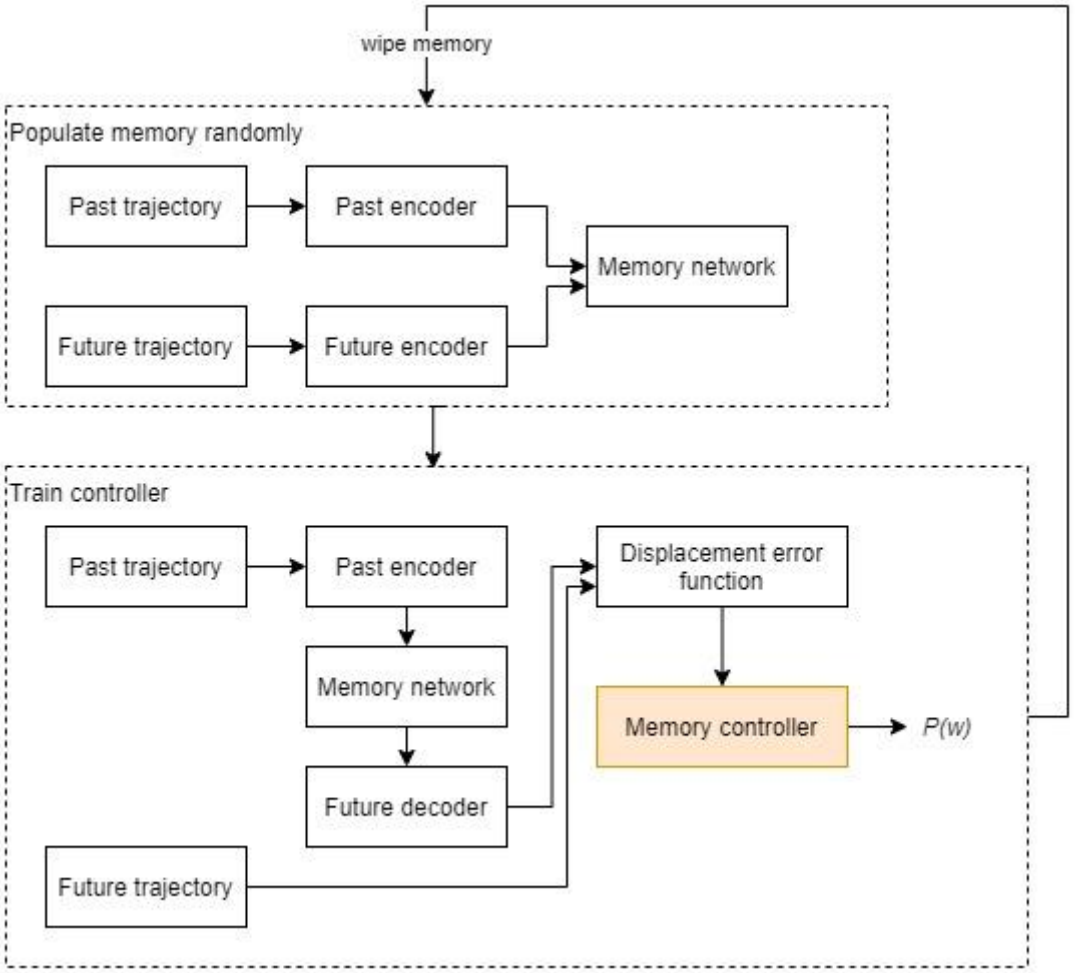

Figure 19. Diagram of pretraining 2

In pretraining 2, we trained the memory controller. The process of training memory controllers consists of three steps depicted in Figure 19. First, we started with an empty memory network and populated the memory network with a random size of data picked from the dataset. Then, we had a memory network that we could work with for the next step. The second step was to train the controller. The past trajectory went to the past encoder, then the memory network will query the best encoded future trajectory according to cosine similarity from Eq. 4. Then the encoded future trajectory prediction is moved to the decoder and we have a future trajectory prediction. Using displacement error function (Eq. 6.) and future trajectory ground truth we calculated the error *e*. After that, the error went to the memory controller, and the model returned the writing probability *P(w)*. Then the *P(w)* finally went to loss function (Eq. 6.) and went to backward propagation. Finally, we repeated the process after wiping the memory. The process is iterated until the maximum epoch is reached or it has already converged.

Finally, we describe the final training phase. The final training is as described in the global architecture in section 5.2. However, the final training consists of two steps. The first step is to populate the memory using the trained memory controller from the previous pretraining so that the memory will not be filled with redundancies. The second step is to train the model to predict the future trajectory given the past trajectory and its map. The model that is trained in model-1 is depicted in Figure 12, while model-2 is in Figure 14.

5. Experimental Result

This chapter consists of six sections. The first section covers the experimental setup. The dataset description is presented in the second section. The results of the experiments conducted with model-1 are presented in the third section. The fourth section presents the experimental result with model-2. We present comparative results in the fifth section. The sixth section presents some discussion on the results.

5.1. Experiment Setup

The specification server we used is a computing node with two Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz. And we used 32 Gigabytes of RAM. We also provided a 1TB HDD. Lastly, for the main computing power, we used NVIDIA Tesla V100 GPUs with 32 Gigabyte VRAM. For training and testing one model, we spend about 10 hours.

In terms of the evaluation metrics, we provide our findings using two standard criteria for predicting vehicle trajectory: Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE is the average error across all future timesteps, and FDE is the error at the corresponding timestep. Therefore, ADE 1s means the average from 0 to 1s errors, while FDE means only the error at the given second. In the results, $k = 1$ represents unimodal cases, and $k = 5$ represents multimodal cases. As in [13], [19] we select the best model based on the performance in the multimodal cases.

ADE and FDE can more formally be represented as follows:

$$ADE = \frac{1}{N}\sum_{i=1}^{N} |\hat{x}_{f_i} - x_{f_i}| \tag{7}$$

$$FDE = |\hat{x}_{f_N} - x_{f_N}| \tag{8}$$

where N is the number of future steps, $\hat{x}_{f_i}$ is future trajectory prediction at timestamp *i*, and for x_{f_i} is the ground truth at timestamp *i*.

5.2. Dataset

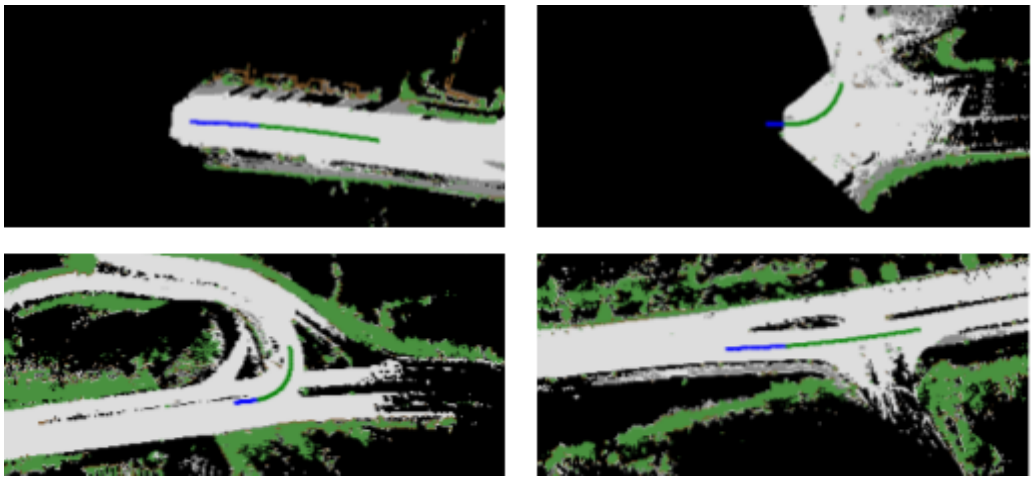

Figure 20. Kitti dataset sample images

In this work, the publicly available dataset Kitti is used [3] for experimentation. Kitti is the same dataset that is used by MANTRA [13]. The original dataset includes many properties such as Velodyne LiDAR 3D scans, bounding boxes and tracks, calibration, and depth. However, we already got it preprocessed into BEV images from [13]. The dataset is split into 2 seconds past trajectories and 4 seconds future trajectories, where every second means ten timeframes. Therefore the dataset is split into 20 frames of past and 40 frames of future positions. For model training, we used a 75:25 ratio or 8613 top-view trajectories for training and 2907 for testing. The dataset was also split exactly the same as in [13]. The dataset sample is shown in Figure 20, where blue is the past trajectory and green is the future.

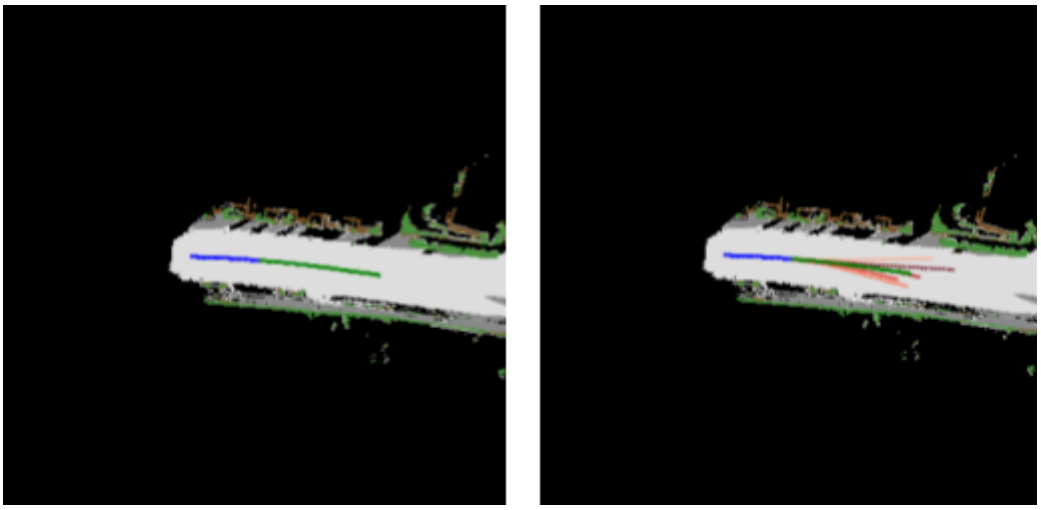

Figure 21. Overview of the inputs(left) and outputs(right).

Table 6. Dataset sizes

Name	Size
Input	
Past Trajectory	2×20
Future Trajectory	2×40
Map Input	$4\times244\times244$
Output	
Future Trajectory Prediction	$2\times40\times k$

The sample of inputs and outputs of the model side-by-side can be seen in Figure 21. The inputs consist of three parts. The first part is the past trajectory represented with two dimensions, x and y coordinate times 20, the length of the timeframe of the past trajectory. The past trajectory is represented with the color blue. The second input is the future trajectory, where it is two dimensions times 40, the length of designated future frames. The last input is the context image. It is represented with a 4-channels RGBA image with width and height are both 244 pixels. The inputs and outputs sizes are presented in Table 6.

5.3. Model-1 Hyperparameter Result

In this section, we present the hyperparameter result experiment for model-1. This is the model where the self-attention layer is positioned before the decoder. The experiments focused on finding the best configuration for the multimodal case because as mentioned in the introduction, the future of the trajectory prediction is multimodal.

As mentioned previously, the number of heads and layers will be discovered in this section. In order to get the best combination of them, we conduct the experiment with two steps. The first step is to find the optimal number of layers for self-attention. We tried a different number of layers, with a number of heads of 8. We decided to pick that number since the number of default heads in the original paper [10] is 8. The second step is to pick the best layer from the previous step and try different combinations of heads and pick the best result from there.

Table 7. Hyperparameter result on model-2 on $k = 5$ where the head is 8

Heads	Layers	FDE				ADE			
		1s	2s	3s	4s	1s	2s	3s	4s
8	1	0.362	0.895	1.721	2.960	0.211	0.424	0.724	1.135
8	**2**	**0.343**	**0.841**	**1.625**	**2.828**	**0.202**	**0.401**	**0.681**	**1.072**
8	4	0.351	0.861	1.661	2.880	0.213	0.413	0.699	1.098
8	8	0.363	0.866	1.687	2.979	0.212	0.415	0.705	1.117
8	16	0.345	0.872	1.723	2.985	0.213	0.413	0.709	1.125

The result of the first hyperparameter experiment is depicted in Table 7. As we can see the least errors for all parameters of both FDE and ADE from 1 second to 4 seconds are achieved when the head's size and layers are both consecutively 8 and 2. The best result for FDE at the 4 seconds is 2.828 meters and for ADE at the same second is 1.072 meters. Based on this experiment we see that more layers do not necessarily give better performance, but only one layer is not optimal.

Table 8. Hyperparameter result on model-2 on $k = 5$ where the head is 8

Heads	Layers	FDE				ADE			
		1s	2s	3s	4s	1s	2s	3s	4s
1	2	0.379	0.883	1.704	2.898	0.234	0.438	0.726	1.123
2	2	0.367	0.894	1.719	2.969	0.217	0.426	0.724	1.136
4	2	**0.339**	0.852	1.684	2.925	**0.19**6	**0.394**	0.687	1.097
8	2	0.343	0.841	1.625	2.828	0.202	0.401	**0.681**	1.072
16	2	0.385	**0.823**	**1.604**	**2.772**	0.243	0.433	0.697	**1.071**

For the second hyperparameter experiment, the result is presented in Table 8. Unlike the previous experiment, we do not have superior results that are the best at every second. The best result on FDE and ADE at 1 second is where the heads are 4. Meanwhile, for the FDE and ADE at the last second (4th seconds), the best result is from the heads of 16. The best result for FDE and ADE at the last second is 2.772 meters and 1.071 meters. Since we favor the longer seconds rather than shorter ones because that means the model can make better anticipation by having a longer time window, we decided to go with the heads of 16 and layers of 2.

5.4. Model-2 Hyperparameter Result

Similar to the previous section, here we present the hyperparameter result on model-2. This section follows the same steps as the previous section. First, we find the best layer size by experimenting with different numbers of layers with the same head's size. The result of the first step is presented in Table 9. After that, we find the most favorable number of heads by trying different sizes of the head.

Table 9. Hyperparameter result on model-2 on k=5 where the head is 8

Head	Layer	FDE				ADE			
		1s	2s	3s	4s	1s	2s	3s	4s
8	2	0.397	0.878	1.570	2.748	0.226	0.439	0.702	1.069
8	3	0.408	0.905	1.604	2.779	0.232	0.453	0.721	1.090
8	4	**0.383**	**0.832**	**1.494**	**2.654**	**0.222**	**0.424**	**0.672**	**1.021**
8	8	0.401	0.873	1.531	2.694	0.229	0.440	0.698	1.054
8	16	0.384	0.853	1.548	2.729	0.224	0.430	0.689	1.053

From Table 9 we can see the best configuration where the error is the minimum is where the layer is 4. The result is superior to other combinations because it is the best for every second on both ADE and FDE. The best FDE and ADE results consecutively are 2.654 meters and 1.021 meters. Similar to the previous experiment, the deeper layers do not necessarily mean better performance.

Table 10. Hyperparameter result on model-2 on k=5 where the layer is 4

Head	Layer	FDE				ADE			
		1s	2s	3s	4s	1s	2s	3s	4s
1	4	0.393	0.870	1.532	2.656	0.226	0.435	0.692	1.045
2	4	0.406	0.879	1.555	2.730	0.233	0.442	0.702	1.065
4	4	0.407	0.890	1.572	2.710	0.233	0.450	0.712	1.071
8	4	**0.383**	**0.832**	**1.494**	**2.654**	**0.222**	**0.424**	**0.672**	**1.021**
16	4	0.396	0.848	1.523	2.675	0.227	0.432	0.686	1.041

The next step is finding the optimal head size. We can see the result from Table 10. Apparently, the best result is the same as the previous experiment, which is 8. That showed that the more parallelization of self-attention does not mean the error will be minimal. Therefore we can conclude the best hyperparameter result is where the head's size is 8 and layers size is 4 and both FDE and ADE are 2.654 meters and 1.021 meters.

5.5. Comparative Result

In this section, we provide comparative results from baselines, MANTRA, and our models. The baseline models presented in this section are Kalman filter, linear network, and multi-layer perceptron (MLP) from the literature [13]. These baseline models are unimodal models. For MANTRA, MAEMANN-1 (model-1), and MAEMANN-2 (model-2) we provide unimodal as well as multimodal results, where multimodal means five modes.

Table 11. Comparative experiments result

Model	FDE				ADE			
	1s	2s	3s	4s	1s	2s	3s	4s
Kalman	0.97	2.54	4.71	7.41	0.51	1.14	1.99	3.03
Linear	0.40	1.18	2.56	4.73	0.20	0.49	0.96	1.64
MLP	0.40	1.17	2.39	4.12	0.20	0.49	0.93	1.53
MANTRA *(k = 1)*	0.68	2.01	4.11	7.02	0.33	0.84	1.60	2.60
MANTRA *(k = 5)*	0.39	0.93	1.78	3.10	0.22	0.45	0.75	1.18
MAEMANN-1 *(k = 1)*	0.480	1.425	2.897	4.929	0.251	0.604	1.134	1.843
MAEMANN-1 *(k = 5)*	**0.385**	0.823	1.604	2.772	0.243	**0.43**3	0.697	1.071
MAEMANN-2 *(k = 1)*	0.683	1.991	4.026	6.707	0.337	0.845	1.580	2.547
MAEMANN-2 *(k = 5)*	0.383	**0.832**	**1.494**	**2.654**	**0.222**	0.424	**0.672**	**1.021**

We put forward all results in Table 11. The MANTRA models used here are computed on our server with the maximum epoch of 600, that is why the result is slightly different from the original paper [13]. For the MAEMANN-1, for both $k = 1$ and $k = 5$, we used the best configuration we got as discussed in section 5.3. For the MAEMANN-2 (i.e. for $k = 1$ and $k = 5$) the hyperparameter configuration is also as discussed earlier in section 5.4.

From Table 11, we can see that a physic-based trajectory prediction [6] like a Kalman filter is not enough for trajectory prediction with an error of 7.41 meters in prediction. However MANTRA and our models on unimodal cases where $k = 1$ are also not significantly better. MANTRA, MAEMANN-1, MAEMANN-2 results on FDE at the 4 seconds are 7.02 meters, 4.929 meters, and 6.707 meters. Even Linear and MLP models on unimodal cases are better than our models with FDE 4.73 meters and 4.12 meters. However, in multimodal cases, our models outperform all the baselines. For multimodal cases MANTRA, MAEMANN-1, and MAEMANN-2 are the FDE at the 4s are 3.10 meters, 2.772 meters, and 2.654 meters. From all three MANN-based models, the number of cases that are peeked from the memory is tied to the number of modes. That means in unimodal experiments those models only queried one case from the memory and this led to bad performances. This shows that only picking one memory from the memory network is not a wise choice.

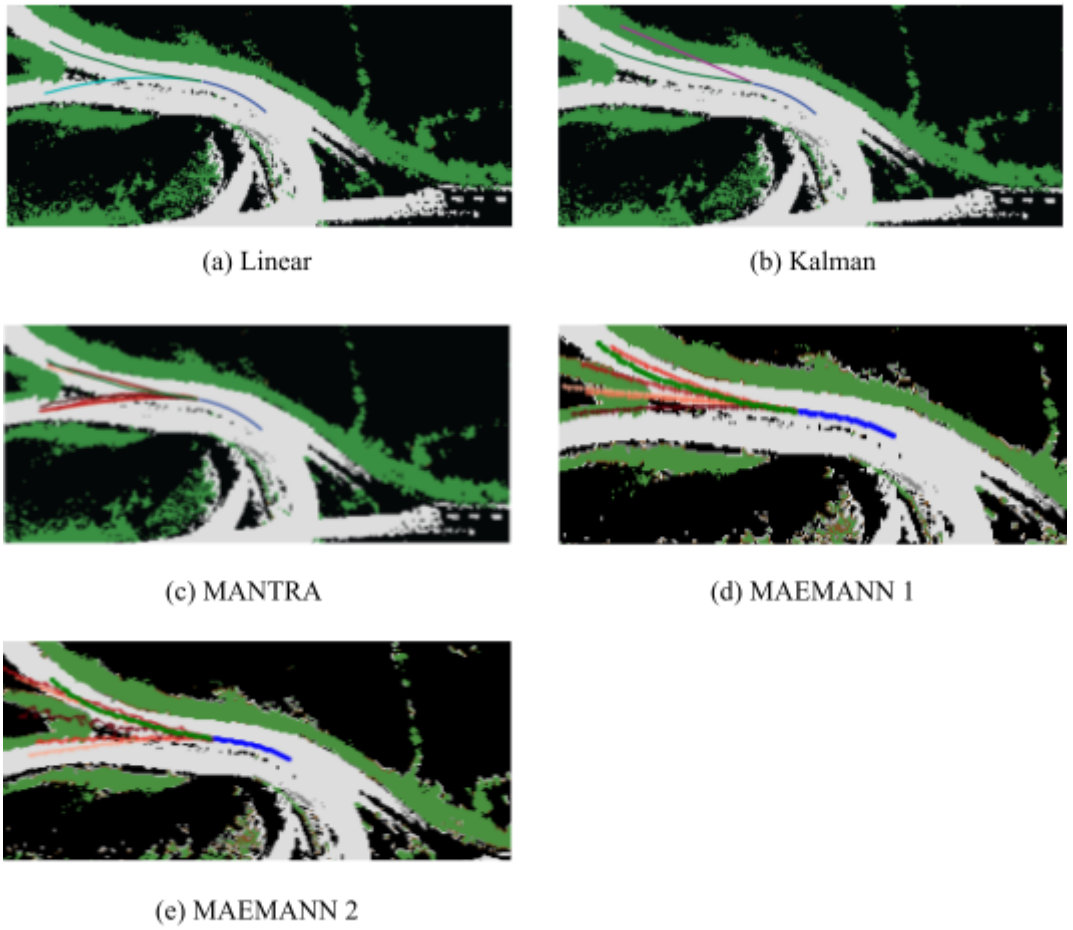

Figure 22. Comparison prediction of (a) linear model [13], (b) kalman filter [13], (c) MANTRA [13], (d) MAEMANN-1, (e) MAEMANN-2

In Figure 22, a sample from the linear model, Kalman filter, MANTRA, MAEMANN-1, and MAEMANN-2 is shown. From the figure it can be seen that the Kalman filter model, based only on physical features of the input, makes a wrong prediction i.e. going straight through the side of the road. Meanwhile, the linear model performs better by following the shape of the road, however, its unimodality fails to predict equally good possibilities for the vehicle to follow different directions of the road. For the MAEMANN-1, some predictions are good, but some others go outside of the road, which means that the model somehow thinks that it is a wise decision to go in the middle between two options of the road. However, the MAEMANN-2 shows visually similar behavior to MANTRA.

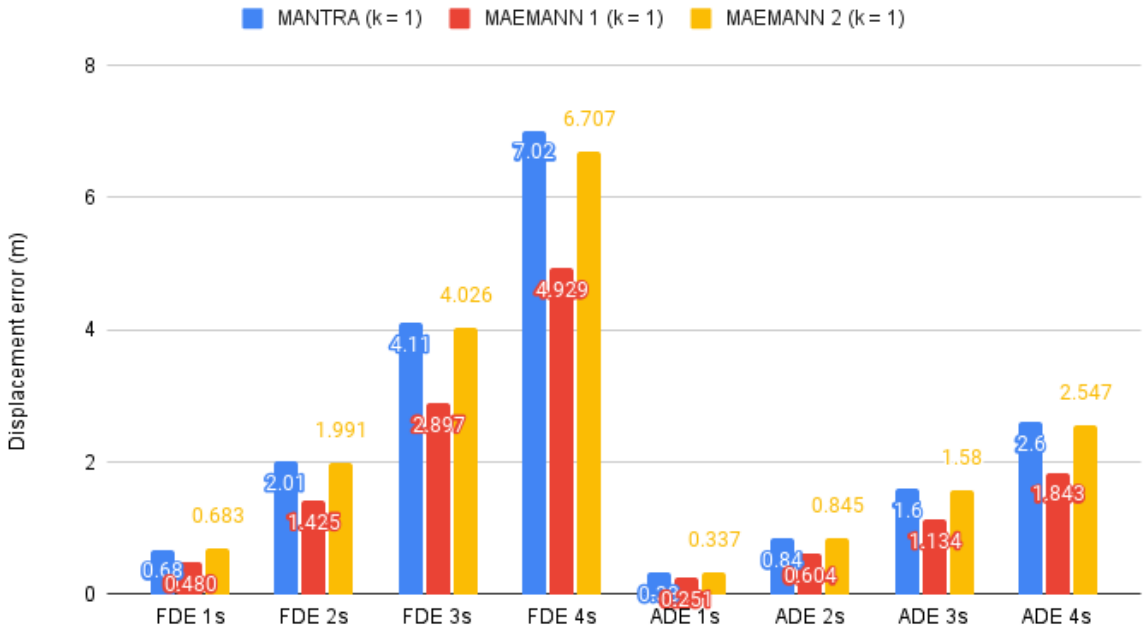

Figure 23. Comparison result for unimodal cases where all the models have $k = 1$. The y-axis is in meters. The x-axis from left to right are FDEs and ADEs at the corresponding second. The smaller the error means a better model.

From Figure 23 and Table 11 we can see the comparison results for MANTRA, model-1, and model-2. As it can be observed in the comparison, our models outperform MANTRA in both ADE and FDE from 1 to 4 seconds. Moreover, the MAEMANN-1, Multi-Head Attention (MHA) enhancement on memory network, showed the best performance of all MANN-based models on unimodal cases. model-1 improves the prediction results in comparison to the original MANTRA model in terms of the FDE metric by 29.79% and ADE by 29.12%. For model-2, the FDE improves by 4.46% and ADE by 2.04%.

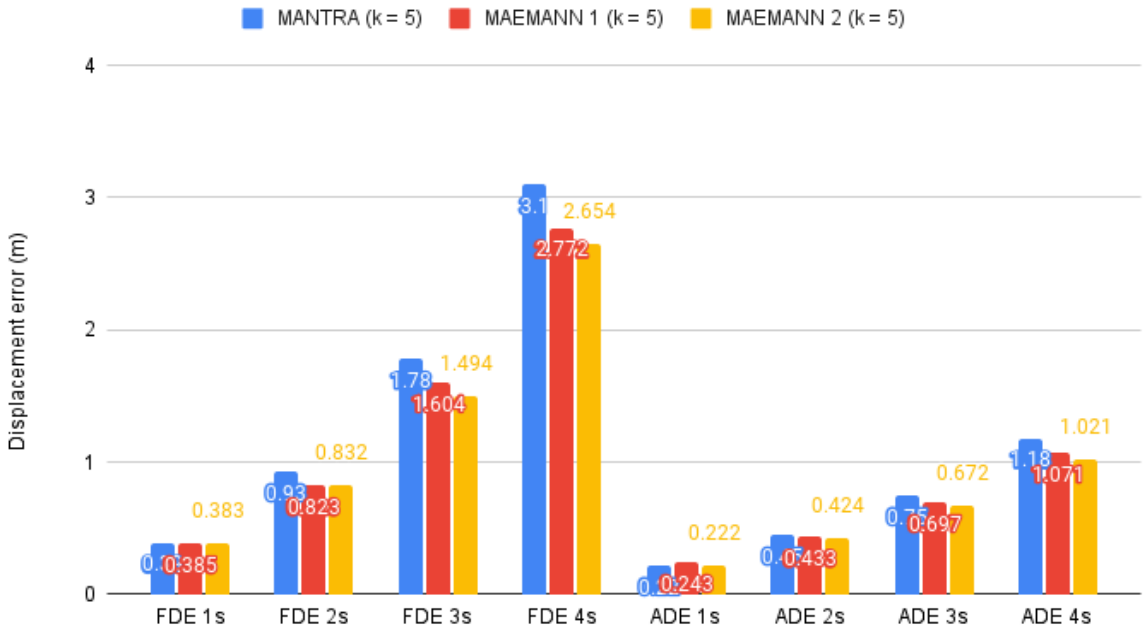

Figure 24. Comparison result for multimodal cases where $k = 5$. The y-axis is in meters. The x-axis from left to right are FDEs and ADEs at the corresponding second. The smaller the error means a better model.

The experimental results for multimodal cases are presented in Figure 24. It can be seen from the figure that our two MAEMANN models outperform MANTRA. In contrast to the unimodal case, in the multimodal case model-2 performs better than model-1. model-1 outperforms MANTRA, in terms of FDE by 10.58 % and in terms of ADE by 9.24 %.. Meanwhile, model-2 outperforms MANTRA in terms of FDE by 14.39 % and in terms of ADE by 13.47 %.

5.6. Discussion

In the hyperparameter experiments (section 5.3 and section 5.4), we investigate the best combination of head and layer for model-1 and model-2. For model-1, the experiments reveal 8 for the head's size and 4 for the layer's size as most suitable. For model-2, the best head size is 16 and the number of layers is 2. From both the experiments, we conclude that the bigger the head size does not mean better performance. The same conclusion was drawn for layers, deeper attention layers does not always reduce the error.

After the hyperparameter experiment, we did a comparison performance between our models, MANTRA and baseline models. The result is that our models outperform the former model and baseline models. However, adding an attention layer where it gives attention to neighboring queried memories in the MAEMANN-1 needs further evaluation since further inspection shows that the model possibly gets distracted by other trajectories resulting in some prediction to go outside of the road. For MAEMANN-2 since the attention mechanism is on the IRM, the model behaves like MANTRA but with better results.

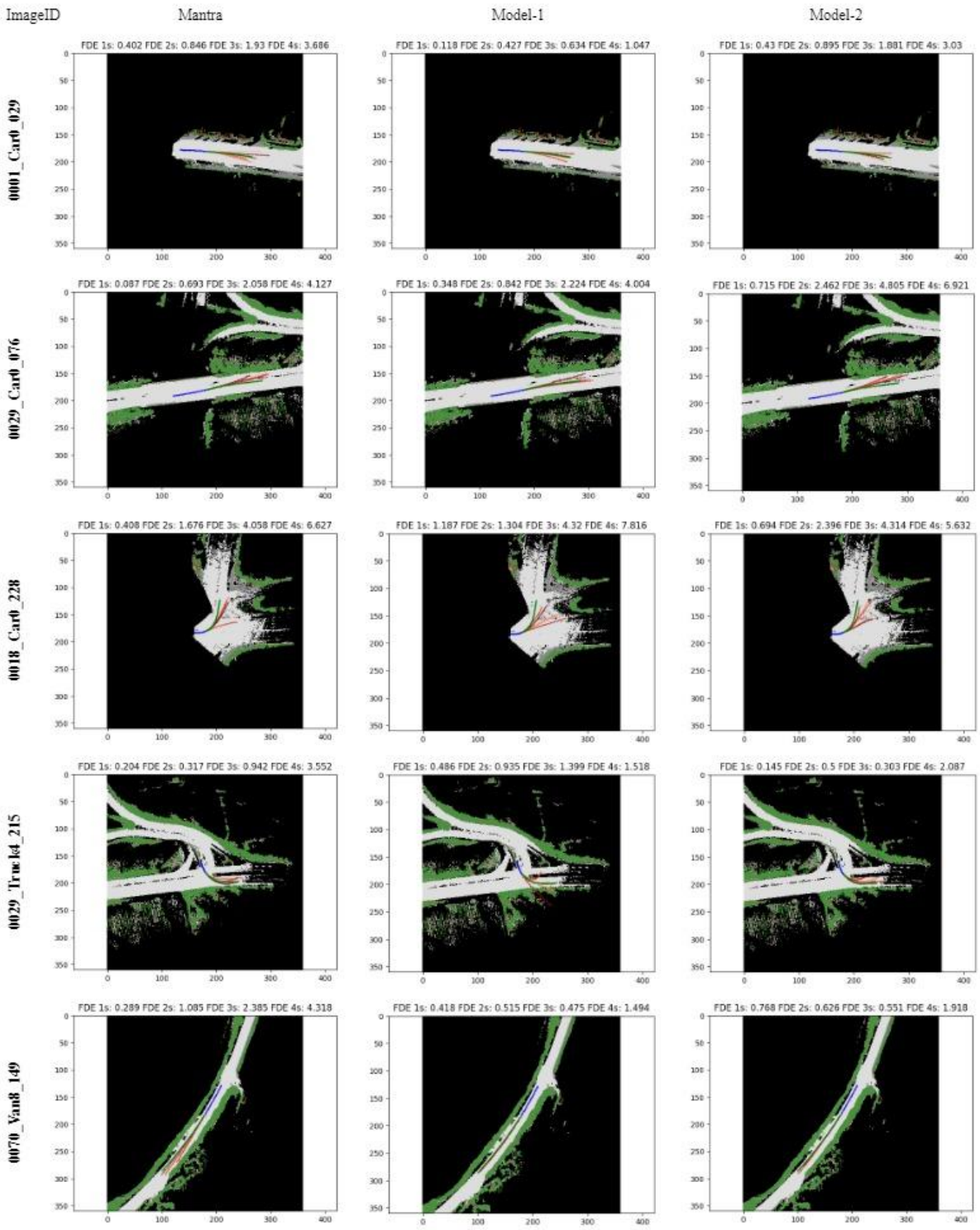

Figure 25. Five randomly chosen example trajectory prediction instances using MANTRA (left), model-1 (mid), and model-2 (right). Each row represents a trajectory prediction instance with the three different prediction models. The grid units are in meters.

Figure 25. show performance of our models and MANTRA on a few randomly chosen (i.e. not hand-picked) trajectories. The figure consists of multiple images where each row represents the same timeframe scene with the prediction performed using different models, named by image ID. From left to right the models are MANTRA, MAEMANN-1, and

MAEMANN-2. From the first row(i.e. image 0001_Car0_029), we can see that the model-1 gets significantly better than the other models. One apparent reason for this is that in the simple case like a straight road, observing multiple memories does not make the model confused, instead it gives the model better accuracy in which trajectory the TV is going to go.

The second row in Figure 25 (i.e. 0029_Car0_076), shows a similar result (in comparison to the results of 0001_Car0_029). The model-1 outperformed other models. This might be because of the simple road layout which model-1 is good at. Also in this row there is an anomaly where MANTRA performs better than model-2. As it can be observed, almost in all cases except this case, model-2 outperforms the MANTRA because it has better IRM in general.

The third row (i.e. 0018_Car0_228), in Figure 25 is a hard case of trajectory prediction, and all models produce large displacement errors. However, the model-2 got the best result. It might be due to better attention to the surrounding environment. Meanwhile the model-1 performs poorly, this might be related to the ability of the model to handle complex road layout.

The fourth row (i.e. 0029_Truck4_215) is also a complex road layout. From the picture we can see MANTRA and model-2's results are visually similar, but model-2 had a better result. However, regarding model-1, some predictions are out of track, but the result is good in general. This is unusual considering model-1 performs badly on complex layout. In our opinion this might be an outlier event for model-1.

In the last row (i.e. 0070_Van8_149), as expected from a straight road, model-2 performs better than MANTRA because it has a better IRM module. Model-1 is significantly better because this case has a simple road layout.

To conclude Figure 25, somehow the model-1 looks really good in our random sample pick. However, aggregate results presented in section 5.5 show the model-2 performs better. Also generally, model-2 is really similar to the MANTRA model because it has the same memory network as MANTAR, the difference is in the IRM module. However, in model-1, there is a significant difference in prediction due to modification in the memory network, but the model still gets distracted somehow and runs outside the road layout.

We did not experiment with higher modes of 10 and 20 (which was done in MANTRA [13]) because, in most competitions, to the best of our knowledge, the multimodality in trajectory prediction is less than 10. For example, in the Lyft trajectory prediction competition [5] the number of multimodality is 3. In the Argoverse competition [49] the number of modes is 6. In addition to that, the multimodal evaluation gives an advantage on more number of modes, because it only picks the best guess out of k number of guesses, which gives the model more chances to try more guesses.

Self-attention has great potential in trajectory prediction because of its attention mechanism that enhances the importance between one feature and another, and adding such a mechanism to a model could improve its performance. It has also been proven earlier in multiple contexts such as NLP [18]–[22] and computer vision [51]–[54]. Moreover, based on our experiments, it is proved that the self-attention mechanism improves the performance of MANTRA. Not only that, self-attention has excellent flexibility and can be placed in different places of the models.

We also want to mention some other models that we experimented with, but did not perform as well as we had intuitively thought them to. We tried to combine model-1 and model-2 but the result was not as positive as them running separately. In our understanding, the reason for this is because creating a deeper model does not necessarily improve the models.

We experimented to replace the autoencoder with MHA, and performed another experiment with a full transformer. As we explained, our models add MHA enhancement, but in this case, we remove the RNN and replace it with our layer instead. However, the results are not favorable. This indicates that RNN is important to make an excellent encoder and decoder.

6. Conclusion and Future Works

This chapter consists of two sections. The first section concludes this thesis. Some possible future works are outlined in the second section.

6.1. Conclusion

In this thesis, we explored the concept of memory augmented neural networks (MANN) on predicting the trajectory of road agents in the context of autonomous driving. We based our implementation on the existing MANN-based trajectory prediction model MANTRA [13]. We developed two enhancement models on the MANN-based model using the self-attention mechanism. The first model or model-1 is a multihead-attention enhancement on the decoder of the model, while the second model or model-2, is a multihead-attention enhancement on the Iterative Refinement Module (IRM).

For model-1, query results from memory networks merge together and then fed to the self-attention layer. This layer will output the attention-enhanced result to pass on to the next layer. This enhancement proved to be able to outperform the original MANTRA model on the furthest seconds, in terms of FDE from 3.1 meters to 2.772 meters - or a 10.58 % improvement. ADE improved from 1.18 meters to 1.071 meters, or in other words a 9.24 % improvement.

In model-2, we put the self-attention mechanism inside the IRM, positioned after the decoder and before the RNN. This implementation also outperformed MANTRA model, in terms of FDE at the 4 seconds, from 3.1 meters and 2.654 meters, or in other words an improvement of 14.39 %. For ADE calculations, the improvement the model makes (in comparison to MANTRA) is from 1.18 meters to 1.021 meters, or an improvement of 13.47 %.

These results prove that the self-attention mechanism has great potential for trajectory prediction in autonomous driving. Same as the self-attention, the MANN-based model also holds significant promise along with a lot of room for improvement. The MANN-based models also provide a unique approach on solving machine learning problems that hopefully in the future more researchers will investigate .

6.2. Future Works

There are many different ways, experiments, and tests that have been left for future work because of lack of time since the implementation and experiments require a lot of time. Some of the worth mentioning directions are as follows:

- **Testing of MANN-based models on a different dataset.** It would be interesting to try MANTRA, MAEMANN-1, and MAEMANN-2 on another dataset rather than the Kitti. For example, the Lyft dataset [5] and the Argoverse [4]. However, the adjustment in implementation is needed because they have different inputs compared to the Kitti [3].

- **Testing different autoencoders.** In this thesis, we used an RNN-based autoencoder where both encoder and decoder are GRU. In [50], they used a transformer based autoencoder so there is potential there.

- **Putting input map to the autoencoder**. In this thesis, our memory network only used the history of TV as an input. We do not use map input except in IRM. So, that means our memory network does not utilize map input. Therefore, it might be a good

idea to utilize map input for the memory network instead of only using the history of TV.

- **Try to implement 1 mode to N number of queries.** In this thesis the number of modes is tied to one query, making it a 1-to-1 mode to query. Another possibility is to implement a 1-to-n mode to query and make the decoder consider multiple queried memories instead of only one.
- **Explore the interpretability of the memory produced by MANN.** Interpretability in machine learning is gaining popularity in the research community. Therefore exploring this direction gives unique value since the MANN model has memory and interpretability of the memory, to the best of our knowledge, is still left unexplored.
- **Output different trajectory predictions types.** Some other trajectory prediction output types like occupancy grid can be investigated.
- **Try our enhancement method on other models that use IRM.** Since MAEMANN-2 improves the performance only by modifying the IRM. Then intuitively self-attention could be used to improve the performance of other models that utilize IRM, for example, DESIRE [19].

7. Reference

[1] S. Singh, “Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey,” *Traffic Saf. Facts - Crash Stats*, Art. no. DOT HS 812 115, Feb. 2015, Accessed: Sep. 14, 2021. [Online]. Available: https://trid.trb.org/view.aspx?id=1346216&source=post_page

[2] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, “A Survey of End-to-End Driving: Architectures and Training Methods,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2020, doi: 10.1109/TNNLS.2020.3043505.

[3] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[4] M.-F. Chang *et al.*, “Argoverse: 3D Tracking and Forecasting with Rich Maps,” *ArXiv191102620 Cs*, Nov. 2019, Accessed: Oct. 29, 2021. [Online]. Available: http://arxiv.org/abs/1911.02620

[5] J. Houston *et al.*, “One Thousand and One Hours: Self-driving Motion Prediction Dataset,” *ArXiv200614480 Cs*, Nov. 2020, Accessed: Oct. 22, 2021. [Online]. Available: http://arxiv.org/abs/2006.14480

[6] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *ROBOMECH J.*, vol. 1, no. 1, p. 1, Jul. 2014, doi: 10.1186/s40648-014-0001-z.

[7] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, “Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review,” *IEEE Trans. Intell. Transp. Syst.*, pp. 1–15, 2020, doi: 10.1109/TITS.2020.3012034.

[8] A. Zyner, S. Worrall, J. Ward, and E. Nebot, “Long short term memory for driver intent prediction,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1484–1489. doi: 10.1109/IVS.2017.7995919.

[9] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” *ArXiv14061078 Cs Stat*, Sep. 2014, Accessed: Oct. 29, 2021. [Online]. Available: http://arxiv.org/abs/1406.1078

[10] A. Vaswani *et al.*, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2017, pp. 6000–6010.

[11] Z. Zhang, P. Cui, and W. Zhu, “Deep Learning on Graphs: A Survey,” *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2020, doi: 10.1109/TKDE.2020.2981333.

[12] J. Weston, S. Chopra, and A. Bordes, “Memory Networks,” *ArXiv14103916 Cs Stat*, Nov. 2015, Accessed: Sep. 23, 2021. [Online]. Available: http://arxiv.org/abs/1410.3916

[13] F. Marchetti, F. Becattini, L. Seidenari, and A. Del Bimbo, “MANTRA: Memory Augmented Networks for Multiple Trajectory Prediction,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 7141–7150. doi: 10.1109/CVPR42600.2020.00717.

[14] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” May 2020. Accessed: Sep. 19, 2021. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv200514165B

[15] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, “Transformer Networks for Trajectory Forecasting,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, Jan. 2021, pp. 10335–10342. doi: 10.1109/ICPR48806.2021.9412190.

[16] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, “AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting,” Mar. 2021. Accessed: Sep. 19, 2021.

[Online]. Available: https://ui.adsabs.harvard.edu/abs/2021arXiv210314023Y
[17] B. T. Morris and M. M. Trivedi, "A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1114–1127, Aug. 2008, doi: 10.1109/TCSVT.2008.927109.
[18] A. Zyner, S. Worrall, and E. Nebot, "A Recurrent Neural Network Solution for Predicting Driver Intention at Unsignalized Intersections," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1759–1764, Jul. 2018, doi: 10.1109/LRA.2018.2805314.
[19] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2165–2174. doi: 10.1109/CVPR.2017.233.
[20] D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, "Generalizable intention prediction of human drivers at intersections," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1665–1670. doi: 10.1109/IVS.2017.7995948.
[21] N. Deo and M. M. Trivedi, "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 1179–1184. doi: 10.1109/IVS.2018.8500493.
[22] Y. Hu, W. Zhan, and M. Tomizuka, "Probabilistic Prediction of Vehicle Semantic Intention and Motion," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 307–313. doi: 10.1109/IVS.2018.8500419.
[23] X. Li, X. Ying, and M. C. Chuah, "GRIP: Graph-based Interaction-aware Trajectory Prediction," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 3960–3966. doi: 10.1109/ITSC.2019.8917228.
[24] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-Head Attention for Multi-Modal Joint Vehicle Motion Forecasting," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 9638–9644. doi: 10.1109/ICRA40945.2020.9197340.
[25] T. Zhao *et al.*, "Multi-Agent Tensor Fusion for Contextual Trajectory Prediction," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 12118–12126. doi: 10.1109/CVPR.2019.01240.
[26] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "HOME: Heatmap Output for future Motion Estimation," May 2021. Accessed: Oct. 25, 2021. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2021arXiv210510968G
[27] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, May 2018, pp. 2056–2063. doi: 10.1109/ICRA.2018.8460874.
[28] M. Schreiber, S. Hoermann, and K. Dietmayer, "Long-Term Occupancy Grid Prediction Using Recurrent Neural Networks," in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, May 2019, pp. 9299–9305. doi: 10.1109/ICRA.2019.8793582.
[29] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A Survey of End-to-End Driving: Architectures and Training Methods," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2020, doi: 10.1109/TNNLS.2020.3043505.
[30] J. Gao *et al.*, "VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 11522–11530. doi: 10.1109/CVPR42600.2020.01154.
[31] S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan, "What-If Motion Prediction for Autonomous Driving," *ArXiv200810587 Cs Stat*, Aug. 2020, Accessed:

Oct. 26, 2021. [Online]. Available: http://arxiv.org/abs/2008.10587
[32] M. Liang *et al.*, “Learning Lane Graph Representations for Motion Forecasting,” *ArXiv200713732 Cs*, Jul. 2020, Accessed: Oct. 26, 2021. [Online]. Available: http://arxiv.org/abs/2007.13732
[33] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *ROBOMECH J.*, vol. 1, no. 1, Art. no. 1, Jul. 2014, doi: 10.1186/s40648-014-0001-z.
[34] H. Cui *et al.*, “Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks,” in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 2090–2096. doi: 10.1109/ICRA.2019.8793868.
[35] “Lyft Motion Prediction for Autonomous Vehicles.” https://kaggle.com/c/lyft-motion-prediction-autonomous-vehicles (accessed Oct. 09, 2021).
[36] M. Gulzar, Y. Muhammad, and N. Muhammad, “A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving,” *IEEE Access*, vol. 9, pp. 137957–137969, 2021, doi: 10.1109/ACCESS.2021.3118224.
[37] Y. Karaca, C. Cattani, and M. Moonis, “Comparison of Deep Learning and Support Vector Machine Learning for Subgroups of Multiple Sclerosis,” in *Computational Science and Its Applications – ICCSA 2017*, Cham, 2017, pp. 142–153. doi: 10.1007/978-3-319-62395-5_11.
[38] S. Liu, S. Liu, W. Cai, S. Pujol, R. Kikinis, and D. Feng, “Early diagnosis of Alzheimer’s disease with deep learning,” in *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, Apr. 2014, pp. 1015–1018. doi: 10.1109/ISBI.2014.6868045.
[39] M. Al-Ayyoub, A. Nuseir, K. Alsmearat, Y. Jararweh, and B. Gupta, “Deep learning for Arabic NLP: A survey,” *J. Comput. Sci.*, vol. 26, pp. 522–531, May 2018, doi: 10.1016/j.jocs.2017.11.011.
[40] Y. Gu and G. Leroy, “Use of Conventional Machine Learning to Optimize Deep Learning Hyper-parameters for NLP Labeling Tasks,” Jan. 2020. doi: 10.24251/HICSS.2020.128.
[41] C. Su, Z. Xu, J. Pathak, and F. Wang, “Deep learning in mental health outcome research: a scoping review,” *Transl. Psychiatry*, vol. 10, Dec. 2020, doi: 10.1038/s41398-020-0780-3.
[42] N. Djuric *et al.*, “Uncertainty-aware Short-term Motion Prediction of Traffic Actors for Autonomous Driving,” *ArXiv180805819 Cs Stat*, Mar. 2020, Accessed: Oct. 27, 2021. [Online]. Available: http://arxiv.org/abs/1808.05819
[43] “What Is The Difference Between CNN And RNN?” https://www.telusinternational.com/articles/difference-between-cnn-and-rnn (accessed Oct. 27, 2021).
[44] W. Ding, J. Chen and S. Shen, "Predicting Vehicle Behaviors Over An Extended Horizon Using Behavior Interaction Network," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8634-8640, doi: 10.1109/ICRA.2019.8794146..
[45] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” *ArXiv160902907 Cs Stat*, Feb. 2017, Accessed: Oct. 27, 2021. [Online]. Available: http://arxiv.org/abs/1609.02907
[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” *ArXiv171010903 Cs Stat*, Feb. 2018, Accessed: Oct. 27, 2021. [Online]. Available: http://arxiv.org/abs/1710.10903
[47] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, “Key-Value Memory Networks for Directly Reading Documents,” in *Proceedings of the 2016*

Conference on Empirical Methods in Natural Language Processing, Austin, Texas, Nov. 2016, pp. 1400–1409. doi: 10.18653/v1/D16-1147.

[48] "A Beginner's Guide to Attention Mechanisms and Memory Networks," *Pathmind*. http://wiki.pathmind.com/attention-mechanism-memory-network (accessed Oct. 27, 2021).

[49] "EvalAI: Evaluating state of the art in AI," *EvalAI*. https://eval.ai (accessed Oct. 09, 2021).

[50] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal Motion Prediction With Stacked Transformers," 2021, pp. 7577–7586. Accessed: Nov. 08, 2021. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Liu_Multimodal_Motion_Prediction_With_Stacked_Transformers_CVPR_2021_paper.html

[51] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck Transformers for Visual Recognition," *ArXiv210111605 Cs*, Jan. 2021, Accessed: Jul. 11, 2021. [Online]. Available: http://arxiv.org/abs/2101.11605

[52] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention Augmented Convolutional Networks," *ArXiv190409925 Cs*, Sep. 2020, Accessed: Jul. 11, 2021. [Online]. Available: http://arxiv.org/abs/1904.09925

[53] W.-C. Hung *et al.*, "SoDA: Multi-Object Tracking with Soft Data Association," *ArXiv200807725 Cs*, Aug. 2020, Accessed: Jul. 11, 2021. [Online]. Available: http://arxiv.org/abs/2008.07725

[54] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, and J. Shlens, "Scaling Local Self-Attention for Parameter Efficient Visual Backbones," *ArXiv210312731 Cs*, Jun. 2021, Accessed: Jul. 13, 2021. [Online]. Available: http://arxiv.org/abs/2103.12731

[55] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *ArXiv201011929 Cs*, Jun. 2021, Accessed: Jul. 13, 2021. [Online]. Available: http://arxiv.org/abs/2010.11929

[56] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, "Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction," *ArXiv200508514 Cs*, Jul. 2020, Accessed: Jul. 10, 2021. [Online]. Available: http://arxiv.org/abs/2005.08514

[57] L. L. Li *et al.*, "End-to-end Contextual Perception and Prediction with Interaction Transformer," *ArXiv200805927 Cs*, Aug. 2020, Accessed: Jul. 10, 2021. [Online]. Available: http://arxiv.org/abs/2008.05927

[58] C. Lüscher *et al.*, "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention -- w/o Data Augmentation," *Interspeech 2019*, pp. 231–235, Sep. 2019, doi: 10.21437/Interspeech.2019-1780.

Appendix

I. Glossary

Acronyms	
TV	Target Vehicle
SV	Surrounding Vehicle
MANN	Memory Augmented Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MANTRA	Memory Augmented Networks for Multiple Trajectory Prediction
NLP	Natural Language Processing
IRM	Iterative Refinement Module
BEV	Bird's Eye View
KF	Kalman Filter
HMM	Hidden Markov Model
SVM	Support Vector Machine
LSTM	Long Short Term Memory
GRU	Gated Recurrent Units
GNN	Graph Neural Network

II. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Farhan Syakir,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Multihead Attention Enhanced Memory Augmented Neural Network (MAEMANN) for Multimodal Trajectory Prediction

(*title of thesis*)

supervised by Naveed Muhammad and Yar Muhammad.

(*supervisor's name*)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Farhan Syakir

11/11/2021