Comparative Analysis of Motion Prediction Algorithms for Autonomous Driving

Mohana Prasath Saravanan
Commercial Vehicle Technology
Rhineland-Pfälzische Technische Universität
Kaiserslautern, Germany
kus05jof@rhrk.uni-kl.de

Saniket Sunil Agarkar
Commercial Vehicle Technology
Rhineland-Pfälzische Technische Universsität
Kaiserslautern, Germany
sud7naj@rhrk.uni-kl.de

***Abstract*— This investigation presents a comprehensive analysis of motion prediction algorithms for autonomous cars, highlighting the critical role that these algorithms play in providing safe and effective navigation in complex environments. It includes a wide range of algorithms, including the Kinematic Bicycle Model, Kalman Filter, Constant Acceleration Model, Multi-layer Perceptron (MLP), and Long Short-Term Memory (LSTM) networks. We trained and tested the models based on the provided datasets, and found what are the most crucial features which affect the performance and accuracy of different motion prediction models. The methodology section describes how to use the datasets, what libraries are needed, and how to prepare the data, and the experimental framework goes into detail on the precise training parameters for each algorithm.**

Keywords—Multi-Layer Perceptrons, Neural Networks, Kalman Filter, Long Short Term Memory, Bicycle Model, Constant Acceleration Model

I. INTRODUCTION

Most automobile firms struggle to ensure the safety and dependability of autos [1]. Operating a motor vehicle necessitates that drivers comprehend intricate, multi-actor scenarios instantaneously and respond promptly. With more than 5% of all deaths in the US in 2015 coming from traffic accidents, this was the fourth most common cause of death [2]. 2017 continued to rank among the deadliest years for drivers in the previous ten years, despite investments in traffic safety [3]. Assisting humans in driving could save many lives and billions in accident-related expenses [4], as human error is the cause of up to 94% of crashes [5]. Autonomous vehicles (AVs) can help overcome this issue, but to drive forward without human input AVs need to solve some of the complex scenarios.

According to [1], to decide how to drive forward on their own, AVs must anticipate how the other cars will move in the future. An accurate and quick estimation of the surrounding cars' motion is required to proceed [1]. Although there are other approaches available now for the same issue, it is crucial to have one that is quick, efficient (using less processing power), and capable of anticipating uncertainties that may arise on the roadways [1]. In this research, we have looked at neural network-based techniques that are more appropriate for practical uses [1].

As stated in [6], Numerous methods have been put forth thus far to examine the motion of the vehicles. Trajectory prediction has been done using vehicle motion models, such as kinematic or dynamic models [7][8]. To conduct prediction accounting for the uncertainty in vehicle models, the Kalman filter has been employed extensively [8]. Bayesian filtering approaches like the context-dependent interactive multiple model filter [7] and the Monte-Carlo method [9] have been presented to further improve the prediction accuracy. Machine learning techniques have been used recently to learn the intricate model that captures the aim of the driver's maneuvers and their interactions with the data. Gaussian processes [10] [11]and Gaussian mixture models [12] have been used to learn the vehicle trajectory-generating model. More complex models that considered the interactions between the vehicles were introduced, such as the dynamic Bayesian network [13].

This paper will examine and evaluate the Constant Velocity Model, Constant Acceleration Model, Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Kalman Filter algorithms for trajectory prediction of surrounding vehicles.

The vehicle's trajectory is predicted with constant velocity, constant acceleration, MLP, and LSTM. For the fifth technique, we used the Kalman Filter algorithm to anticipate the future positions of cars, trucks, and buses for 5 seconds. We used the constant velocity model to forecast pedestrian and bicyclist movements, which decreases processing power and time requirements. We anticipate the future position for 0.5 seconds. The average loss is compared to that of other neural network models used for pedestrian position prediction.

The neural network's methodology is described in Section II. Section III provides the terminologies and the datasets. The evaluation of the performance of various techniques is explained in Section IV. Section V discusses the results of the models. Section VI concludes the paper.

II. METHODOLOGY

Vehicle motion prediction can be accomplished through two distinct approaches namely, the Physics-Based Approach and the Data-Driven Approach.

A. Physics-Based Approach:

Using the fundamental ideas and laws of physics, a physics-based approach to motion prediction forecasts the future motion of objects. This method is based on simulating the forces and interactions that influence an object's motion and then using the results to predict an object's behavior over time.

With this method, a mathematical model that mimics the motion of objects is created by taking into account several physical parameters, including forces, velocities,

accelerations, and limitations. Newton's laws of motion, which outline how forces impact an object's motion, are the most frequently applied physical laws in motion prediction. Depending on the particular situation, other laws and concepts from mechanics, electromagnetism, and other pertinent branches of physics may also be included.

According to Z. Mo et al. [14], the shortcomings of physics-based models are mainly attributable to their inability to be generalized since they rely on predetermined motion heuristics that make strong assumptions about driving behaviors based on a limited set of characteristics. The subtleties of human strategic planning behaviors and their flexibility in highly interactive driving settings may be beyond the scope of these heuristics. Consequently, in a variety of authentic driving conditions, these algorithms' prediction ability might be compromised.

Dynamic models and kinematic models are the two types of physics-based models.

1. Dynamic Models

Lagrange's equations, which take into account a variety of factors impacting the vehicle's movement, such as longitudinal and lateral tire forces as well as the road banking angle, are used by R. Rajamani et al. [15]to illustrate vehicle motion in dynamic models. Complex physics, which considers how driver actions affect different components like the engine, gearbox, wheels, and more, influence vehicles that resemble cars. Because of this, dynamic models for these cars can get extremely comprehensive and include a wide range of internal factors that are specific to each car. According to S.Lefèvre [16], for trajectory prediction, simpler models are often preferred. One commonly used representation is the "bicycle" model, which treats a car as a two-wheeled vehicle with front-wheel drive moving on a 2-D plane. Trajectory prediction benefits greatly from this optimized method since it strikes a balance between computational efficiency and accuracy. [16]

Our goal with the kinematic bicycle model is to forecast a vehicle's or bicycle's motion depending on its inputs, like steering angle and velocity[17]. By taking into account the vehicle's kinematics—that is, how it moves without taking into account things like forces and torques—this model offers a condensed description of how the vehicle travels [17].

Source [17] stated that the two primary components of the kinematic bicycle model are the longitudinal dynamics (forward/backward motion) and the lateral dynamics (steering), which are predicated on the assumption that the vehicle moves in a planar (2D) environment. In [17], the following presumptions form the basis of the model:

Longitudinal Dynamics: The car travels at a steady speed or in a straight line. The car ignores its braking and acceleration [17].

Lateral Dynamics: By changing the steering angle, the car steers. The steering angle and the vehicle's velocity determine how far the vehicle will travel laterally [17].

Using the kinematic bicycle model, we usually use numerical integration techniques and differential equations to estimate the vehicle's future motion. We can mimic the trajectory of the vehicle over time by giving inputs like the steering angle and initial velocity. The kinematic bicycle model is helpful for tasks like path planning and trajectory prediction in different applications, including autonomous driving and robotics, even though it is an oversimplified depiction that ignores complex dynamics like tire forces and vehicle mass. It can help in making well-informed decisions on the movement of the vehicle and gives a decent estimate of its motion. [17]

$$x = Tsv \cos(\phi) + x0 \quad (1)$$

$$y = Tsv \sin(\phi) + y0 \quad (2)$$

$$\phi = \phi + (Tsv \tan(\gamma\ 0))\ /L) \quad (3)$$

2. Kinematic Models

By defining mathematical links between motion-related variables including position, velocity, and acceleration, kinematic models explain how a vehicle moves. Interestingly, they don't take into account the forces influencing this motion. These models assume that each wheel's velocity and direction match exactly, without considering friction forces.

According to S. Lefèvre et al. [16], kinematic models are frequently preferred over dynamic models for trajectory prediction tasks due to their suitability for non-vehicle control applications and their ease of use. Kinematic models offer a more direct and useful method of trajectory prediction in situations where vehicle control is not the main concern [16].

Different levels of complexity can be defined to provide a systematic approach to the classification of motion models, according to R. Schubert et al. [18]. Linear motion models, characterized by constant acceleration (CA) or constant velocity (CV), are found at the bottom end of this scale [18]. These models' main benefit is that their linearity permits the state probability distribution to propagate as optimally as possible [18]. These models, however, have limited applicability since they assume linear motions and ignore rotations, especially the yaw rate [18].

A model of predicting trajectories using a motion model with constant velocity is shown in Fig. 1.[16]

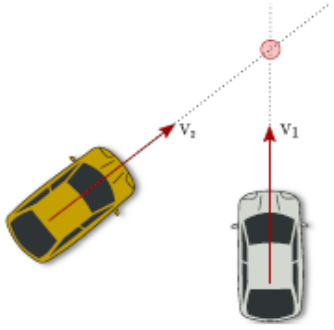

Fig. 1. Constant Velocity Model

Motion models get more detailed as complexity levels rise, capturing a wider variety of vehicle behavior and enabling sophisticated decision-making algorithms in autonomous vehicles [16].

B.Data-Driven Approach:

Data-driven methods include the Multi-Layer Perceptron (MLP) and Recurrent Neural Network (RNN). Recurrent neural networks differ from traditional neural networks in that

they have a unique architecture. Its main goal is to deal with situations in which the data shows a sequential structure, with connections between the elements placed in a specific order.

According to A. Zyner et al [19], these sequences can be encountered in a variety of circumstances, including the word order of a sentence and the temporal ordering of data, like time-series sensor data or video frames. In these situations, a single time step is represented by each step in the sequence. The main goal of using a recurrent neural network is to efficiently simulate the linkages and correlations between the next samples in the series.

1. Long-Short Term Memory:

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to effectively detect and characterize long-term dependencies in sequential data. LSTMs, originally described by Hochreiter and Schmidhuber in 1997, have now become an essential building block in many sequential data processing applications, such as time series analysis, speech recognition, and natural language processing [20]. Memory cells and certain gating mechanisms are the building blocks of an LSTM network, which allows the network to remember or forget information over extended periods [20]. Conventional RNNs cannot handle the vanishing gradient problem, which occurs when training networks to learn long-range dependencies, however, LSTMs can decrease it [20].

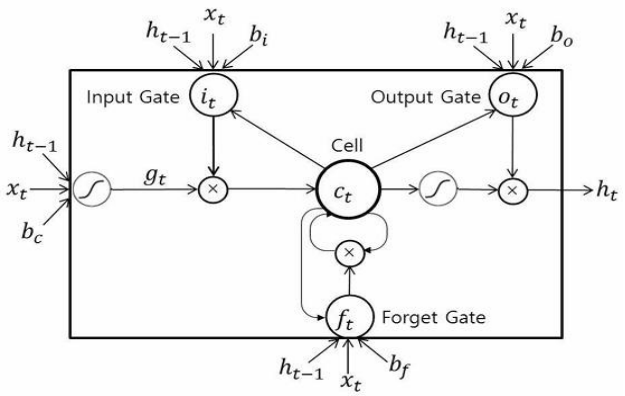

Fig. 2. LSTM Basic Structure [1, Fig.4]

The LSTM network's long-term memory is comprised of memory cells. Because of its ability to store and retrieve data across lengthy periods, the network can continue to store pertinent data even in the face of significant data gaps. In the LSTM architecture, the memory cells are commonly shown as horizontal lines. Specialized gating mechanisms are used by LSTMs to regulate the information flow into and out of the memory cells. What data should be removed from the memory cells is decided by the forget gate. What new data should be stored in the memory cells is decided by the input gate. The information that should be output from the memory cells is determined by the output gate. The LSTM's horizontal cell state maintains the capacity to modify through the gates while enabling information to move across time steps. The forget gate, input gate, and output gate control the state of the cell. Activation functions are commonly used by LSTMs to regulate the output of the gates and memory cells. The hyperbolic tangent (tanh) function and the sigmoid function are the most often utilized activation functions in long short-term memory (LSTMs). Long-term dependencies in sequential data can be captured and stored by LSTMs by selective updating and gate access to the memory cells. The network gains the ability to alter gates and memory cells in response to patterns and dependencies discovered in the data during training.[20]

LSTMs have been a great tool for processing and modeling sequential data in several applications. Significant advancements have been made in areas such as machine translation, sentiment analysis, speech recognition, and handwriting recognition. If you want more in-depth information or specific references on LSTMs, I recommend reading research papers by Hochreiter and Schmidhuber (the authors of the initial article on LSTMs) and later work in the disciplines of deep learning and sequential modeling.

According to A. Zyner et al [19], LSTM successfully solves the problem of disappearing gradients because of its special ability to hold a value for any length of time [19]. Unlike traditional RNNs, which face difficulties during training since input either decreases or increases exponentially over time, LSTM uses a gating method to get around this issue [19].

Recently, long recognized for their prowess in text analysis, LSTMs have demonstrated remarkable abilities in problems involving sequence generation. Because of this recently discovered capacity, LSTMs are now preferred above other varieties of neural networks. Long-standing applications such as speech synthesis and music composition have demonstrated the versatility of LSTMs in producing sequences, despite their traditional employment in text parsing. Their increasing relevance in deep learning is highlighted by their versatility. In conclusion, LSTMs are becoming more and more popular for a variety of sequence creation tasks in addition to text processing.

2. Multi-Layer Perceptron:

It is a feedforward neural network model composed of multiple layers of artificial neurons, or nodes, that are connected and arranged in a sequential manner[1]. A type of artificial neural network (ANN) used in many machine learning applications, such as pattern recognition, regression, and classification, is the multilayer perceptron (MLP) [1].

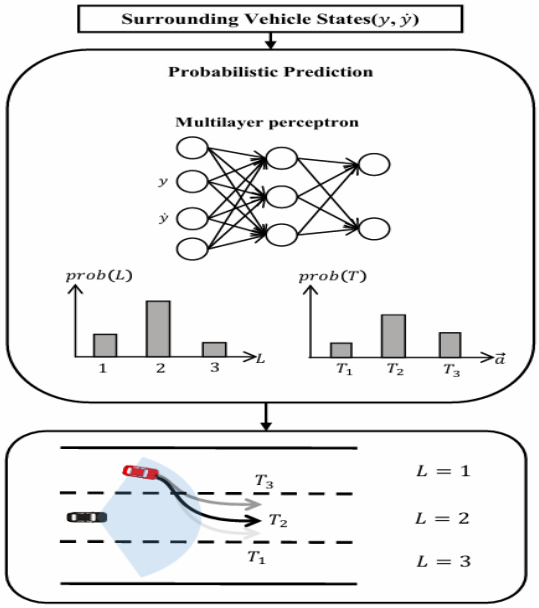

Fig. 3. An overview of the MLP-based probabilistic lateral motion prediction technique [1, Fig.1]

From Fig.1 [1, Fig.1] An example of a neural network having interconnected neurons and a nonlinear activation function is the Multilayer Perceptron, or MLP. It is specifically engineered to process non-linearly separable data [1]. The Multilayer Perceptron (MLP) is widely employed in diverse domains like speech recognition, natural language processing, and picture classification [1]. Since it can approximate any function under specific conditions and has a flexible design, it is regarded as a vital component in deep learning and neural network research [1].

- The input layer is responsible for receiving the initial input data from nodes or neurons. Every individual neuron in the input layer corresponds to a distinct aspect or dimension of the incoming data.

- The hidden layer serves as a connection between the input and output layers. Each neuron in the buried layer receives inputs from all neurons in the previous layer.

- The output layer is responsible for producing the ultimate output of the network. The number of neurons in the output layer changes based on the precise task at hand.

C.Kalman Filter:

The Kalman Filter is an algorithm that estimates the state of a system over time, even with noisy measurements. It works in two steps: prediction and update. First, it predicts the current state based on past data. Then, as new measurements come in, it updates this prediction, balancing the reliability of the model with the new data. The filter iteratively refines its estimate, making it highly effective for real-time tracking in applications like navigation, robotics, and economics. Its strength lies in its ability to provide accurate estimates despite uncertainty or noise in the data.

III.TERMINOLOGIES AND DATASET

A. Hyperparameter
Neural networks have an adjustable set of parameters called hyperparameters that control how the network learns. They provide a way to train, optimize, and change the neural network's dimensions and structure to suit a specific purpose. Here are the hyperparameters that were used.

B. Layers
A neural network is composed of building blocks arranged in vertical stacks referred to as Layers. Within a neural network, these layers can be categorized into three distinct types.

1) Input Layer
The initial layer receives input data and transfers it to subsequent layers.

2) Hidden Layer
This layer, potentially comprised of multiple sub-layers, contributes to the neural network's intricacy and overall performance.

3) Output Layer
Responsible for delivering the final output of the neural network.

C. Nodes
A layer comprises numerous individual units referred to as neurons. Each neuron within the layer accepts input from another neuron processes this input and generates an output.

D. Activation Function
The activation function within a neural network governs the how input is converted into output as it moves from one node to another within a layer. Among the activation functions employed are Rectified Linear Unit (ReLU), Sigmoid, and Softmax.

E. Batch size
The batch size denotes the number of training datasets utilized in a single iteration.

F. Epochs
Epochs refer to the count of complete passes that the entire training dataset undergoes within the neural network.

G. Learning Rate
The learning rate, a hyperparameter, adjusts and controls the size of the step taken during each iteration as the neural network progresses toward minimizing the loss function.

H. Dataset
A dataset file originating from the Intersection Drone Dataset (inD) has been obtained, encompassing 33 distinct recordings sourced from 4 varied road intersection sites across Germany. Each recording has an approximate time duration of 15 minutes. In these recordings, a track ID is allocated to each vehicle present, and various motion parameters of the vehicle are logged for successive time frames until the vehicle exits the recording area. The dataset encompasses the observation of multiple parameters outlined below.

The dataset encompasses the xCenter (Xc) and yCenter (Yc) values, indicative of the coordinates of the vehicle's midpoint. Additionally, the dataset incorporates Heading Angle (Head) Xvelocity (Xv), and Yvelocity (Yv), representing the velocities of the vehicles in the and y dimensions respectively. Likewise, Xacceleration (Xa) and Yacceleration (Ya) denote the accelerations experienced by the vehicles along the corresponding directions. Moreover, the dataset logs the orientation angles of the vehicles along with their dimensions encompassing length and breadth. Furthermore, the dataset records the longitudinal (LoV), (LoA) velocities and accelerations, lateral velocities and accelerations (LaV), (LaA), considering the driving orientations of each vehicle. The parameters inside the tables such as Recording ID (RI), Past Sequence Length (PSL), Future Sequence Length (FSL), Deviated Distance at the Last Point of Prediction (DSL), Mean Squared Error (MSE), Average Displacement Error (ADE) are abbreviated and used in the tables.

a) Setting up Libraries:
When it comes to Python programming, a library is a vast collection of pre-made code modules, functions, classes, and

other resources that are intended to enhance the functioning of software applications. Libraries are created to streamline the software development process and offer easily accessible answers for typical programming problems. They cover a broad spectrum of tasks and domains, including web development, machine learning, data analysis, and numerical calculation. Libraries give programmers the benefit of utilizing well-tested and efficient implementations, which minimizes the time and effort needed for writing from scratch by encapsulating difficult algorithms, data structures, and tools. The programming community's use and availability of Python libraries is a major factor in the increasing productivity and efficiency of software development projects. We used libraries like Pandas, NumPy, Matplotlib, PyTorch Lightning, and Tensorflow in our investigation. Pandas is a robust data manipulation and analysis framework that offers capabilities for handling structured data as well as data structures (like DataFrames). Large, multi-dimensional arrays and mathematical functions are supported by the core NumPy library for numerical computing. Python users may create static, animated, and interactive visualizations with the Matplotlib charting toolkit. A lightweight PyTorch wrapper or framework called PyTorch Lightning makes training and organizing PyTorch models easier. The goal of PyTorch Lightning is to improve the readability, modularity, and scalability of PyTorch code, especially for large-scale projects and research projects. It allows you to concentrate on model creation and experimentation by streamlining the development process and encouraging best practices.

b) Input Data:
Our study made use of the inD dataset. The inD dataset is a newly released database of realistic car trajectories that were taken at German intersections. With the use of a drone, it gets around the drawbacks of conventional traffic data collection techniques and minimizes problems like occlusions. The dataset comprises recordings from four different sites, from which the trajectories of every category of road user have been retrieved. It is usually possible to limit the positioning error to less than 10 centimeters by using sophisticated computer vision techniques. This dataset can be used by researchers for a variety of purposes, such as driver modeling, scenario-based safety validation of automated driving systems, road user prediction, and data-driven component development for highly automated driving systems.

c) Pre-processing the data:
To pre-process our dataset, we employed the pandas package. All 32 of the inD datasets were used. We normalized these datasets once they were imported. One common preprocessing step in data analysis and machine learning activities is normalizing the information. It entails transforming the data into a normalized scale, ensuring that the magnitude of various variables is comparable. Data normalization is particularly important when the scales of the elements differ. Relevant features were retrieved from the datasets during preprocessing to aid in further analysis and modeling. After that label encoding was done, which is a technique used in machine learning to convert categorical data into numerical form. In label encoding, each unique category in a dataset is assigned a unique integer value.

d) Preparing Data for Training and Testing the Model:
This was a step that we only needed to apply to data-driven models. The data-driven models in our scenario are Long Short Term Memory (LSTM) and multi-layer perceptrons. Following the preparation of the data, the corresponding algorithms receive it and the prediction models are trained.

IV. EXPERIMENTS AND RESULTS

A. Constant Acceleration (Physics Based) Model:

The goal of the single-input, single-output constant acceleration model is to forecast an object's position at a given time given its initial position, initial velocity, and constant acceleration. We assume this model that the object's acceleration doesn't change as it moves. The object's position at any given moment can be determined using the equation if we know its initial position, velocity, and acceleration. In our application of the constant acceleration model for motion prediction, we employed the general acceleration equation:

$$X = ut + 0.5at^2 \quad (4)$$

for both the lateral (y) and longitudinal (x) directions. Using this equation, we can determine the object's or vehicle's position at any given time t by taking into account its initial velocity (u), constant acceleration (a), and the amount of time that has passed. We may anticipate the object's or vehicle's velocity clearly and consistently by applying this equation in both directions, taking into account acceleration effects in both the longitudinal and lateral dimensions.

TABLE I: CA RESULTS

No	RI	PSL	FSL	Test Loss
1	19	6	3	0.02527
2	20	6	3	0.02482
3	21	6	3	0.01924
4	22	6	3	0.02003
5	23	6	3	0.01871

We tested different recording IDs shown in TABLE I, in that recordings 21,22, and 23 show relatively better performance with lower test loss compared to recordings 19 and 20. After performing different combinations of features, the constant acceleration performed well with Xc, Yc, Head, Xv, Yv, LoV, and LaV. As a result, these are considered the most crucial parameters for the Constant Acceleration Model.

B. Bicycle Model:

The constant acceleration model and the constant velocity model are comparable. We supplied heading angles, y velocity, and x velocity as the function's parameters as inputs to the model. Given that the camera was operating at a frame rate of 25 frames per second, the time step Ts was regarded as constant. Therefore, 0.04 seconds was the time step. It was determined that the wheelbase L was 1 m. The following formulas were used by the function to forecast the states:

$$\beta = \tan - 1(\text{vx}|\text{vy}) \quad (5)$$
$$\text{x}_{\text{dot}} = \text{vx}\cos(\emptyset) \quad (6)$$
$$y_{dot} = vy\cos(\emptyset) \quad (7)$$
$$\emptyset_{dot} = (vy.\sin(\beta))/L \quad (8)$$

TABLE II: CV RESULTS (0.12 SECONDS/3 FRAMES)

No	RI	PSL	FSL	Test Loss
1	19	6	3	1.00364
2	20	6	3	1.000929
3	21	6	3	0.81116
4	22	6	3	0.8518
5	23	6	3	0.81307

We ran CV for two different parameters keeping the features such as Xv, Yv, Head, LaV, and LoV as constant as the model performed well with lower loss with these parameters. First, we ran for 0.12 seconds i.e. for 3 frames, we predicted the motion of classes and entered the corresponding test losses as shown in TABLE II. Secondly, we predicted the motion of pedestrians and bicyclists' movements alone for 0.5 seconds i.e. for 13 frames and the results are tabulated in TABLE III. Compared to other Recording IDs, with ID 20 the CV model performed well which can be inferred from the corresponding test loss.

TABLE III. CV RESULTS (0.5 SEC/13 FRAMES)

No	RI	PSL	FSL	(MSE) metres	(ADE) metres
1	19	6	13	0.0018	0.0494
2	20	6	13	0.0047	0.0708
3	21	6	13	0.0040	0.0651
4	22	6	13	0.0019	0.0510
5	28	6	13	0.0010	0.0317

We have plotted the graph of the actual ground truth value versus the predicted values shown in Fig.4. The graph was plotted to predict pedestrians and bicyclists for 0.12 seconds. From the graph, the comparison can seen visually as the predicted path moreover approximates with the groundtruth value which is a good sign that the constant velocity model performed well when it is used for the prediction of movements of pedestrians and bicyclists.

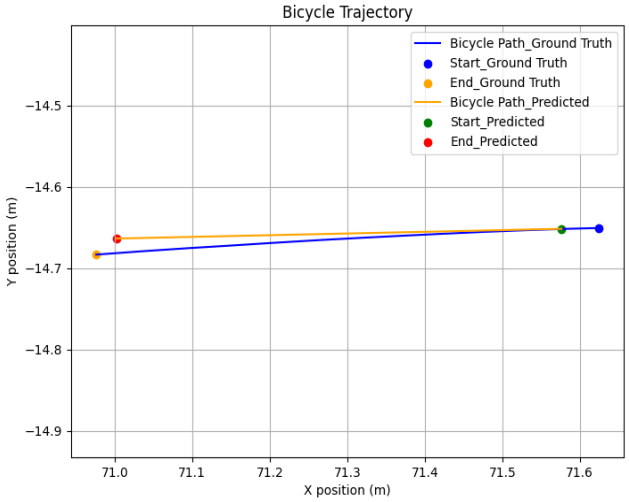

Fig. 4. Ground Truth Vs Predicted Path (CV)

C. Multi-Layer Perceptron:

The MLP model is trained with the following parameters:

The MLP model is designed with an architecture consisting of 32 hidden layers. During training, the data is processed in batches of size 50, and the model undergoes 5, 10, or 20 complete passes (epochs) through the training dataset. The input data has a dimension of the product of several features selected and the past sequence length, while the model produces an output with a dimension of a number of features selected. During the training and testing the value of the past sequence length is 6 and the future sequence length is 3 is kept constant. These parameters define the model's structure and training configuration, ensuring efficient learning and performance.

TABLE IV. MLP RESULTS

No	RI	Epoch	Features	Test Loss	Optimizer
1	19	5	Xc, Yc, Head, Xv, Yv	0.29855	Adam
2	20	20	Xc, Yc, Head, Xv, Yv, LaV, LoV	0.28988	Adam
3	21	20	Xc, Yc, Head, Xv, Yv, LaV, LoV	0.20072	Adam
4	22	20	Xc, Yc, Head, Xv, Yv, LaV, LoV	0.18384	Adam
1_u	01,02,03, 04,05,06, 07,08,09, 10	20	Xc, Yc, Head, Xv, Yv	10.85544	Adam
2_u	01,02,03, 04,05,06, 07,08,09, 10	20	Xc, Yc, Head, Xv, Yv, Xa, Ya,Lv, LaV, LoA, LaA	12.86323	Adam

MLP was trained and tested to predict the motion of all classes for 0.04 seconds with the mentioned features in TABLE IV. MLP predicted the motion better when the Latitudinal (LaV) and the Longitudinal velocities (LoV) are given as input along with the features Xc, Yc, Head, Xv, Yv which is evident from the results of the recording IDs 19,20,21 and 22. In our training and testing, we observed the most crucial features for the MLP are the above-mentioned features.

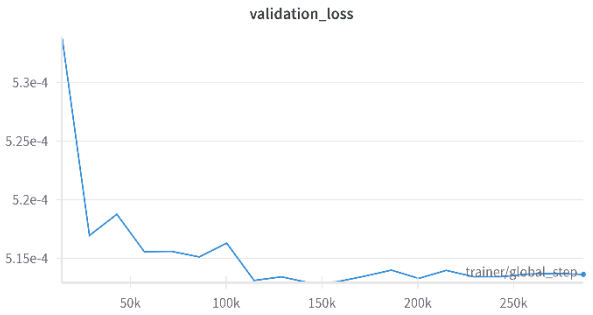

Fig. 5. Validation Loss vs Global Step (MLP)

D. LSTM Recurrent Neural Network:

Our analysis of LSTM (Long Short-Term Memory) models for motion prediction revealed that the most efficient option for our predictive tasks was a more straightforward architecture with a single layer and an Adam optimizer. This finding emphasizes how crucial optimizer choice and model simplicity are to maximizing motion prediction accuracy.

TABLE V. LSTM RESULTS

No	RI	Epoch	Features	Test Loss	Optimizer
1	19	10	Xc, Yc, Head, Xv, Yv	0.3409	Adam
2	20	10	Xc, Yc, Head, Xv, Yv	0.19836	Adam
3	21	10	Xc, Yc, Head, Xv, Yv	0.25791	Adam
4	22	10	Xc, Yc, Head, Xv, Yv	0.28621	Adam
5	23	10	Xc, Yc, Head, Xv, Yv	0.31225	Adam

We used LSTM to predict the motion of all the classes for 0.12 seconds. The minimum test loss is 0.3409 for the recording ID 19 out of 32 Recording IDs provided. We found out after several tests, the features Xc, Yc, Head, Xv, and Yv were the essential key features when compared to the other features such as Lav, LoV, LaA, and LoA.

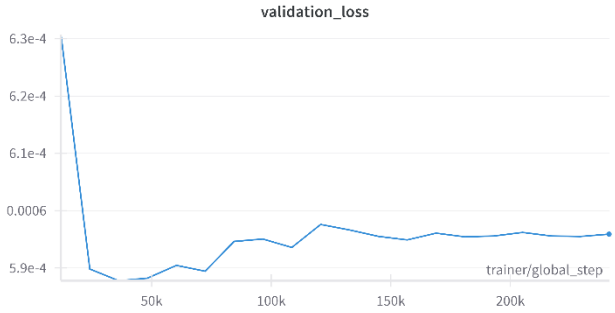

Fig. 6. Validation Loss vs Global Step (LSTM)

The validation loss curve in Fig.6 shows that the LSTM model quickly learns in the early stages but reaches a plateau after a certain number of training steps. As a result, there's no significant overfitting, but further performance improvements may require adjustments to learning rate schedules.

E. Kalman Filter:

The steps for implementing the Kalman Filter are as follows:

1)Prediction: Using the system's dynamics, estimate the next state and its uncertainty.

2)Measurement Update: Incorporate a new measurement, calculate the difference between the predicted and actual state, and adjust the state estimate and its uncertainty based on the measurement's reliability.

3)Iterate: Repeatedly perform the prediction and measurement update steps to refine the state estimation as new measurements become available, providing an optimal estimate of the true state. We assumed the data is obtained from an external source (eg. GPS) and is noisy. We will know the initial location of the agent. Then using the systems dynamics to estimate the next state.

TABLE VI. KALMAN FILTER RESULTS

No	RI	Features	FSL	MSE (metres)	DSL (metres)	ADE (metres)
1	19	Cars, Trucks_Bus, Bicycle, Pedestrians	3	0.0006	7.86E-06	0.00255
2	21	Cars, Trucks_Bus, Bicycle, Pedestrians	3	1.93E-08	5.02E-05	3.15E-05
3	23	Cars, Trucks_Bus, Bicycle, Pedestrians	3	2.59E-09	2.30E-05	1.76E-05
4	26	Cars, Trucks_Bus, Bicycle, Pedestrians	3	1.02E-08	1.08E-05	2.05E-05
5	19	Cars, Truck_Bus	125	1.70E-07	0.00038	0.00033
6	20	Cars, Truck_Bus	125	5.86E-08	0.000288	0.00028
7	22	Cars, Truck_Bus	125	3.67E-08	0.00022	0.00022
8	25	Cars, Truck_Bus	125	3.92E-08	0.0002307	0.0002278

9	27	Cars, Truck_Bus	125	2.85E-07	0.000256	0.00031

Kalman filter was used in two cases. One for predicting all the classes for 0.12 seconds and in another case for predicting the truck/buses and cars for 5 seconds i.e. for 125 frames. The calculated values are tabulated in the TABLE VI.

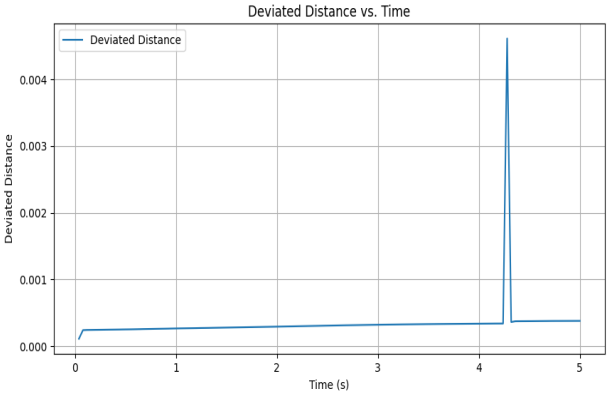

Fig. 7. Deviated Distance vs Time Graph

V. DISCUSSION

The research findings offer a significant understanding of how different motion prediction models function when applied to diverse object categories, such as pedestrians, and vehicles such as trucks, buses, and bicycles. Along with the Kalman Filter method for predicting the locations of cars, trucks, and buses, as well as the constant velocity model for predicting the positions of pedestrians and bicycles, the study assessed physical models (constant velocity, constant acceleration) and data-driven models (MLP, LSTM).

a)Physics-based models:

The performance of physics-based models showed stable and comparatively low error rates across different object types. For instance, the Constant Velocity model attained an MSE of around 0.001 for all classes, indicating its effectiveness for objects exhibiting predictable or less complex movement, such as bicyclists and pedestrians. The marginal difference in error between the Constant Velocity and Constant Acceleration models suggests that, in many real-world scenarios, simpler models may perform adequately for motion prediction tasks.

b) Machine Learning Based Models:

Long-short-term memory and multi-layer perceptron models are utilized in machine learning-based approaches, wherein the errors produced by the two models trained for comparable hyperparameters are essentially the same. The MLP model's minimum test loss is 0.18384, whereas the LSTM model's is 0.19386. After five epochs, both models converge, requiring less processing power and having the capacity to take into account most uncertainties occurring in the real-time environment.

VI. CONCLUSION

In conclusion, our analysis revealed from the findings of this project, that the Kalman Filter outperformed every other model mentioned. The data-driven models such as MLP and LSTM configured with the tuned hyperparameters performed well with low errors. The Constant velocity predicts the movements of pedestrians and bicyclists precisely compared to the other models.

In a nutshell, our research paper presents a comprehensive comparative analysis of various predictive modeling techniques for vehicle motion tracking and prediction. It evaluates the performance of Long Short-Term Memory (LSTM) and Multi-Layer Perceptron (MLP) models, alongside two key kinematic models: the Constant Acceleration Model and the Kinematic Bicycle Model. In addition, the study integrates the Kalman Filter as a well-known filtering method. By systematically assessing these models using real-world driving data, the paper investigates their predictive accuracy, adaptability to different scenarios, and computational efficiency. The research findings offer valuable insights into the strengths and limitations of these models and provide practical recommendations for their application in areas such as autonomous driving, traffic management, and robotics. Moreover, in future the following such as the potential of hybrid models, advanced data preprocessing techniques, hardware acceleration, and uncertainty estimation methods to improve prediction accuracy and reliability can be done to improve the efficiency of the model which could be useful in real time. Future research endeavors can delve deeper into optimizing model architectures, exploring additional data sources, and considering dynamic and complex driving scenarios to further enhance predictive capabilities.

VI. REFERENCES

[1] 2016 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2016.

[2] M. Speaks, “Health United States Report 2016,” 2016.

[3] N. Highway Traffic Safety Administration and U. Department of Transportation, “Crash Stats: Early Estimate of Motor Vehicle Traffic Fatalities for the First Half, January to June of 2017,” 2017.

[4] L. Blincoe, T. R. Miller, and E. Zaloshnja, “Date 6. Performing Organization Code 7. Authors,” 2015. [Online]. Available: www.ntis.gov.

[5] N. Highway Traffic Safety Administration and U. Department of Transportation, “TRAFFIC SAFETY FACTS Crash • Stats Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey.”

[6] B. Kim et al., “Probabilistic Vehicle Trajectory Prediction over Occupancy Grid Map via Recurrent Neural Network,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.07049

[7] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. J. Dietmayer, “IMM object tracking for high dynamic driving maneuvers,” in IEEE Intelligent Vehicles Symposium, Proceedings, 2004, pp. 825–830. doi: 10.1109/ivs.2004.1336491.

[8] S. Ammoun and F. Nashashibi, “Real time trajectory prediction for collision risk estimation between

vehicles." [Online]. Available: https://inria.hal.science/inria-00438624

[9] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using Monte Carlo sampling," in IEEE Transactions on Intelligent Transportation Systems, Mar. 2008, pp. 137–147. doi: 10.1109/TITS.2007.909241.

[10] Intelligent Vehicles Symposium proceedings, 2014 IEEE : date 8-11 June 2014. IEEE, 2014.

[11] C. Laugier et al., "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," IEEE Intelligent Transportation Systems Magazine, vol. 3, no. 4, pp. 4–19, Dec. 2011, doi: 10.1109/MITS.2011.942779.

[12] . IEEE Staff, 2012 IEEE Intelligent Vehicles Symposium. IEEE, 2012.

[13] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," IEEE Intelligent Transportation Systems Magazine, vol. 7, no. 1, pp. 69–79, Jan. 2015, doi: 10.1109/MITS.2014.2357038.

[14] Z. Mo, R. Shi, and X. Di, "A physics-informed deep learning paradigm for car-following models," Transp Res Part C Emerg Technol, vol. 130, Sep. 2021, doi: 10.1016/j.trc.2021.103240.

[15] F. F. Ling et al., "Editor-in-Chief." [Online]. Available: http://www.springer.com/series/1161

[16] S. Lefèvre, D. Vasquez, and C. Laugier, "REVIEW Open Access A survey on motion prediction and risk assessment for intelligent vehicles," 2014. [Online]. Available: http://www.robomechjournal.com/content/

[17] F. Wei, Z. Zhang, and K. Jia, "Research on Kalman filter for one-dimensional discrete data," in Journal of Physics: Conference Series, IOP Publishing Ltd, Aug. 2021. doi: 10.1088/1742-6596/2005/1/012005.

[18] R. Schubert, E. Richter, and G. Wanielik, "Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking."

[19] P. Karle, M. Geisslinger, J. Betz, and M. Lienkamp, "Scenario Understanding and Motion Prediction for Autonomous Vehicles - Review and Comparison," Oct. 01, 2022, Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/TITS.2022.3156011.

[20] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Comput, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[21] C. Montella, "The Kalman Filter and Related Algorithms: A Literature Review The Kalman Filter and Related Algorithms A Literature Review." [Online]. Available: https://www.researchgate.net/publication/236897001