Optimal motion prediction model for car objects

Group 15: Neha Walunjkar - 425760, Pratij Jadhav - 425666, Shubham Sonawane - 425247
Department of Electrical Engineering and Information Technology
Rhineland-Palatinate Technical University of Kaiserslautern-Landau,Kaiserslautern, Germany

Abstract—**In a variety of fields, such as autonomous driving and advanced driver assistance systems, accurate motion prediction for multiple objects, including vehicles and we as vulnerable road users, is of utmost importance. The crucial challenge of choosing the best motion prediction models for this diverse group of entities is addressed in this paper. Our study digs into a comparison of data-driven and physics-based methodologies to identify the best modelling approach for car objects. We investigate the data-driven and physics-based models in depth, and then empirically assess how well they capture the complex dynamics and behaviours displayed by the car objects. We use Average Displacement Error as performance indicators to measure each model's predicted accuracy while drawing on inD dataset. With the use of this large dataset, our study carefully analyses the physics-based and data-driven approaches to provide relevant information about the effectiveness of each strategy for various entities. Our research shows that car objects have a higher level of compatibility with data-driven approaches, making use of machine learning's power to infer patterns from the data. On the other hand, physics-based models perform better in situations with short time horizon.**

Index Terms—**Vehicle motion Prediction, data-driven models, physics-based models, Kalman filter, machine learning.**

I. Introduction

Accurately anticipating the movement of various objects, from vehicles to vulnerable road users, is crucial in today's environment, particularly in applications like autonomous driving. Finding the models that are most effective at forecasting how cars will travel is a major difficulty that is addressed in this research study. We aim to determine the optimal model for accurately predicting the trajectory of car objects of the track ID: 25, considering 1 second into the future for physics-based models, whereas 2 seconds for data-driven models. This involves running data-driven and physics-based models on every track ID and subsequently comparing their performance across each track ID.

Section II of this paper talks about physics-based models, section III presents data driven models, section IV talks about Kalman filter based Constant Velocity model, section V presents fundamentals and results, lastly, section VI talks about conclusion and future work.

A. Information about the dataset

For our motion prediction task, we have used "The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections" as it provides a full depiction of road user dynamics and caters to wide range of research analysis and development endeavours. For each recording instance, the dataset architecture includes a single aerial image, three CSV files, and a digital map. This aerial image shows a drone-captured aerial perspective of the intersection. The initial two CSV files contain vital metadata about the recording and the tracked road users. This metadata includes information such as recording and location ID, frame rate, duration, road user categories, etc. The third CSV file contains detailed trajectory information, including position in X and Y direction, orientation, velocity, and acceleration data for each frame. The inD dataset contains 13,599 road user trajectories, including 8,233 vehicles (cars, trucks, and buses) and 5,366 Vulnerable Road Users such as pedestrians and bikers. The dataset also includes the trajectories of these different road users.[3]

B. Mathematical models for motion prediction

Mathematical models for motion prediction are classified into three types:

1. Physics-based motion models: The behaviour of a vehicle in terms of its movement and trajectory is governed solely by the fundamental principles of physics. The models assumes that the vehicle travels at a constant speed and maintains a fixed orientation throughout its motion. This means that it provides a simplified representation of a vehicle's motion and doesn't take into consideration more complex factors that might affect real-world scenarios.[1]

2. Interaction-aware motion models: This model goes beyond individual vehicle behaviours and considers the intricate interactions between the maneuvers of multiple vehicles. It recognizes that the motion of one vehicle can affect and be affected by the actions of others, capturing the inter dependencies between their trajectories.[1]

3. Maneuver-based motion models: This model takes into account that the future motion of a vehicle is influenced by the intended driver maneuver. These advanced models acknowledge that a vehicle's trajectory is not solely determined by physics laws, but also by the driver's planned actions.[1]

II. Physics based models

Physics-based motion models are mathematical representations of how vehicles move, taking into account the principles of physics that govern their behaviour. These models consider vehicles as dynamic entities because they can change their motion based on the forces acting upon them. To forecast the future motion of a vehicle, these models consider various factors such as control Inputs, vehicle properties, and external conditions. Control inputs are the actions taken by the driver or automated system to control the vehicle's movement. Examples of control inputs include steering the wheel to change direction and applying acceleration or braking to change speed. Properties include specific characteristics, such as its weight, which significantly affect how it responds to the applied forces. Heavier vehicles may have different motion patterns compared to lighter ones due to their mass. The models also take into account external factors that influence the vehicle's motion. One crucial external condition is the friction coefficient of the road surface. The amount of friction between the tires and the road affects how the vehicle accelerates, decelerates, and turns. By considering these inputs, properties, and external conditions, the physics-based motion models can predict how the vehicle's state will change over time. This enables to predict how a vehicle will behave in various driving scenarios, which is crucial for tasks like trajectory planning, collision avoidance, and overall vehicle control. Kinematic models and dynamic models can be roughly categorised as physics-based models in the context of predicting vehicle motion.[1]

Dynamic models: Dynamic models are mathematical representations of the motion of a moving object that take into account a variety of forces acting on the object. Due to how driver movements affect the engine, gearbox, and wheels, car-like vehicles entail sophisticated physics that require huge, intricate dynamic models with a variety of internal characteristics. These complex models are appropriate for control-based activities, whereas simpler models like kinematics models are favoured for trajectory prediction.[1]

Kinematics models: Kinematic models depict how a vehicle moves by utilising mathematical relationships between its motion characteristics (such as location, velocity, and acceleration) without taking into account the forces influencing the motion. These models ignore friction forces and assume that the velocity at each wheel is parallel to the orientation of the wheel. Kinematic models are more widely employed than dynamic models for predicting a vehicle's trajectory since they are simpler and usually sufficient for applications other than control tasks. The Constant Velocity (CV) and Constant Acceleration (CA) models are simple kinematics models that include independent vehicle states representing yaw angle and yaw rate in state vector. Since they are independent, these models are less complex. 'Bicycle model' is also a kinematic model that considers steering angle into the state vector of the vehicle and represents the relationship between velocity and yaw rate of the vehicle.[1]
We are using two physics-based motion prediction models: 1. Constant Acceleration model 2. Bicycle model

A. Constant acceleration model

The constant acceleration model assumes that the acceleration will remain constant between updates as it plots the location, velocity, and acceleration of a moving object through time. The model is based on the fundamental principles of physics, which involve concepts of acceleration, velocity, and position. The constant acceleration model oversimplifies real-world vehicle behaviour. Vehicles do not maintain constant acceleration over long periods, and other factors such as road conditions, traffic, and driver behaviour can significantly impact motion. The model assumes constant acceleration between updates. If updates are infrequent or not synchronized with real changes in acceleration, predictions may deviate from reality. Since real-world vehicle behaviour involves gradual transitions between acceleration, deceleration, and constant velocity, the constant acceleration model may lead to abrupt changes that do not match actual motion. For long-term predictions, the constant acceleration model becomes less accurate as the assumptions of constant acceleration over time intervals break down. Many times, a certain motion model is assumed in order to forecast trajectory. The model assuming Constant Yaw Rate and Acceleration seems to produce the best outcomes [7]. The design model used in [8] has a constant acceleration phase which gives us the constant acceleration model as:

The state vector $\mathbf{x}_{CA} = (x, y, \psi, v_x, v_y, a_x, a_y)$ where (x, y) is the position, ψ is the yaw angle, (v_x, v_y) is the velocity vector, and (a_x, a_y) is the acceleration vector.

The dynamics are given by:

$$\dot{\mathbf{x}}_{CA} = \begin{bmatrix} v_x + a_x t \\ v_y + a_y t \\ 0 \\ a_x \\ a_y \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ n_\psi \\ 0 \\ 0 \\ n_{a_x} \\ n_{a_y} \end{bmatrix} \tag{8}$$

In [9], a Kalman filter is used to track the dynamic state of the vehicle at every scan. The model in [9] chosen for the state equation update considers a constant acceleration (CA) in both axes, so that the transition matrix of the state equation is:

$$A = \begin{bmatrix} 1 & T & 0.5 \cdot T^2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & 0.5 \cdot T^2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The covariance matrix of the process noise will be $Q = G \cdot q \cdot G^T$ where we assumed a zero mean, white, and Gaussian process noise.

If the vehicle is following the geometry of the road, the calculated future course of the vehicle can also take into consideration the geometry of the road borders. Due to its strong performance and affordable computing cost, the CA model is used to update the vehicle's dynamic state:

$$x = x_0 + U_0 \cdot t + \frac{1}{2} \cdot A \cdot t^2 \tag{8}$$

The CA model is useful for short term predictions. For longer time intervals, the error accumulates over the preceding one to give more deviation.

B. Bicycle model

The bicycle model is a more sophisticated physics-based model used to predict the motion of two-wheeled vehicles like bicycles or motorcycles. As shown in fig.1, it approximates the behaviour of a vehicle by considering it as a simple bicycle with two main wheels and a single track. The bicycle model is based on the assumption that the vehicle can be accurately represented by two rigid bodies connected by a hinge (the steering axis). The model takes into account the vehicle's steering angle and yaw rate (rate of change of heading) to predict its future motion.[2]

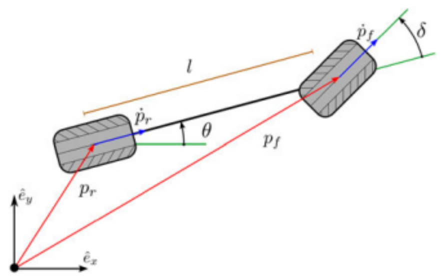


Fig. 1: Bicycle model [2]

The model considers the following main parameters:
Position (x, y): The coordinates of the vehicle's centre of gravity.
Heading (θ): The orientation of the vehicle
Velocity (v): The speed of the vehicle.
Steering angle (δ): The angle of the front wheel with respect to the longitudinal axis of the vehicle.

The basic bicycle model assumes that vehicle has two main wheels: a front wheel and a rear wheel. The front wheel is steered by an angle known as the steering angle (δ) and the vehicle moves in a plane, and its motion is constrained to the longitudinal (forward) and lateral (sideways) directions. By calculating the derivatives of x position, y position and heading angle of bicycle and incorporating them into equations of motion, bicycle model's response to various inputs and external forces, such as steering angles, accelerations, and terrain variations can be modelled and analysed. These derivatives are essential for predicting and controlling the motion of the bicycle in different scenarios.[2]

$$\dot{x} = v\cos(\theta) \tag{2}$$

$$\dot{y} = v\sin(\theta) \tag{2}$$

$$\dot{\theta} = \frac{v}{l}\tan(\delta) \tag{2}$$

III. Data driven models

As shown in fig.2 Data driven deep learning based approaches are classified using three separate criteria i.e. input representation, output type, and prediction method [5].

A. Prediction methods

Deep learning methods for predicting the behaviour of autonomous vehicles are classified into three types: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and alternative deep learning-based methods such as graph neural networks. For prediction methods, we are using LSTM, and vanilla MLP models for motion prediction.

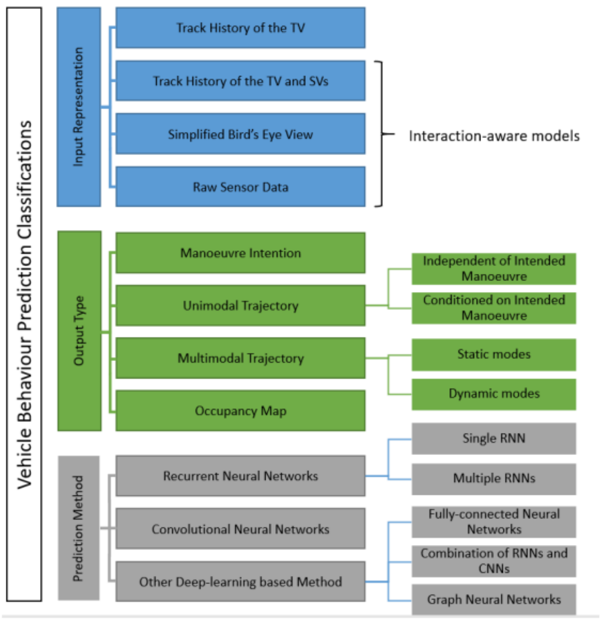

Vehicle Behaviour Prediction Classifications
Input Representation
Track History of the TV
Track History of the TV and SVs
Simplified Bird's Eye View
Raw Sensor Data
Interaction-aware models
Output Type
Manoeuvre Intention
Unimodal Trajectory
Multimodal Trajectory
Occupancy Map
Independent of Intended Manoeuvre
Conditioned on Intended Manoeuvre
Static modes
Dynamic modes
Prediction Method
Recurrent Neural Networks
Convolutional Neural Networks
Other Deep-learning based Method
Single RNN
Multiple RNNs
Fully-connected Neural Networks
Combination of RNNs and CNNs
Graph Neural Networks

Fig. 2: Classifications deep learning approaches for vehicle motion prediction [5]

Vanilla MLP (Multi-Layer Perceptron): An input layer, one or more hidden layers, and an output layer make up a Vanilla MLP. A group of neurons or nodes make up each layer. Information travels through the network in one direction, from the input layer to the output layer, due to its feedforward architecture. A Vanilla MLP's neurons each apply an activation function to the weighted sum of their inputs. The gradients of the loss function with respect to the model parameters are computed via backpropagation, which enables weight updates to reduce the prediction error.[6]

RNN-LSTM: RNN stands for Recurrent Neural Network. They are designed to utilise sequential information. Recurrent neural networks (RNNs) are so named because they carry out the same job for each element of a sequence while encoding past data in an internal state variable hence, RNNs are considered to memory as a result. The vanishing and exploding gradient problems are issues that can occur during the training of Recurrent Neural Networks (RNNs). These problems can make it challenging to train RNNs effectively on long sequences, where information needs to be propagated over many time steps. Long Short-Term Memory (LSTM) networks are a type of RNN architecture designed to mitigate the vanishing and exploding gradient problems. LSTMs address these issues through the use of specialized memory cells and gating mechanisms. They may carefully control the addition or removal of information from state variables using gates. Information can pass through these gates.[4]

B. Input representation

Fig.2 shows that input representation is classified into Track history of Target Vehicle, Track History of Target Vehicle and Surrounding Vehicle, Simplified Bird's Eye View and Raw Sensor Data. For input representation, we are using the Track

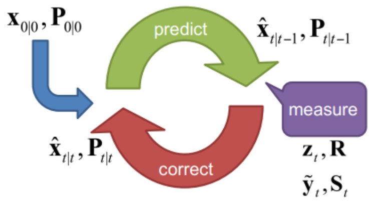
History of the TV (Target vehicle). It is the traditional method for predicting the behaviour of the target vehicle by either tracing the history of its states over time or merely utilising its current state such as position, heading, velocity, etc without considering any interactions. The other three representations refer to Interaction-aware models, which take into account the interactions between different entities in a scene.[5]

C. *Output type*

The prediction model's output categorizes into four classes as shown in fig. 2: maneuver intention, uni-modal trajectory, multi-modal trajectory, and occupancy map. The first class, maneuver intention, foresees the vehicle's planned actions, such as lane changes or turns. The second class, uni-modal trajectory, predicts precise vehicle paths over time, with two subsets: one independent of intended maneuvers, averaging all potential trajectories, and the other conditioned on intended maneuvers, estimating the most likely trajectory based on intention. The third class, multi-modal trajectory, anticipates multiple trajectories for various maneuver intentions, encompassing static modes with predetermined behaviours and dynamic modes adapting to driving scenarios. Lastly, occupancy map methods evaluate cell occupancy in a bird's-eye-view map, rather than predicting trajectories, yet their precision is limited by cell size. Each approach offers unique insights and trade-offs in predicting vehicle behaviour.[5]

Output of our prediction model provides a Uni-modal Trajectory independent of intended maneuvers. It forecasts a sequence of future target vehicle positions over a time interval to characterise how a vehicle will behave in the future.[5]

IV. Kalman filter based Constant Velocity model



$\mathbf{x}_{0|0}, \mathbf{P}_{0|0}$
predict
$\hat{\mathbf{x}}_{t|t-1}, \mathbf{P}_{t|t-1}$
measure
$\mathbf{z}_t, \mathbf{R}$
$\tilde{\mathbf{y}}_t, \mathbf{S}_t$
correct
$\hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}$

Fig. 3: Kalman Filter [11]

The object's position is supposed to change linearly with time in this model, while its velocity stays constant. This indicates that during the time of the prediction, the object is not accelerating or decelerating. By employing a straightforward constant velocity and then fine-tuning it with actual measurements, the Constant Velocity (CV) model and Kalman filter combination enhances tracking accuracy. As shown in fig.3, Kalman Filter(KF) works on the premise that a system's state at time t can be predicted from its state at time step $t-1$. Prediction and Update are two main processes in KF.[10]

$$\theta_t = S\theta_{t-1} + r_t \tag{10}$$

This equation represents the state transition process. It states that the current state θ_t is determined by multiplying the previous state θ_{t-1} by a transition matrix S and adding the process noise r_t which represents random fluctuations or errors. In case of a Constant Velocity model, the state vector typically includes parameters like position and velocity.

$$p_t = H\theta_t + m_t \tag{10}$$

This equation relates the observed measurements p_t to the state θ_t. m_t is measurement noise which represents random errors or noise in the measurements. H is the observation matrix which maps the state space to the measurement space, defining how the state relates to the measurements.

A. *Prediction*

$$\hat{v}_t = S\hat{v}^{(t-1)} + \epsilon_t \tag{10}$$

This equation is part of the prediction step. It predicts the state one step ahead based on the motion model.ϵ_t simulated error at time step t

$$\bar{W}_t = S\bar{W}^{(t-1)}S^T + Q_t \tag{10}$$

This equation is also part of the prediction step. It predicts the error covariance matrix one step ahead.

$$S = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

S is called the transition matrix, where Δt is the time interval between two time steps which is assumed to be constant for simplicity.

B. *Measurement*

$$\hat{v}^t = \hat{v}_t + K_t(p_t - H\hat{v}_t) \tag{10}$$

This equation is part of the measurement update step. It refines the predicted state based on the actual measurements.

$$W_t = (I_4 - K_t H)\bar{W}_t \tag{10}$$

This equation updates the error covariance matrix based on the measurements.

$$K_t = \bar{W}_t H^T (H\bar{W}_t H^T + C_t)^{-1} \tag{10}$$

(K_t) is a weighting matrix that determines how much importance is given to the prediction and measurement.

The future state is predicted by KF using the present state and a mathematical model during the prediction stage. The prediction takes uncertainty noise into account. The filter then updates the estimate based on the accuracy of both the model and the data by combining this prediction with actual measurements. Even in the presence of noise and uncertainties, this adaptive process yields a precise and accurate estimation of the system's state.

V. Fundamentals and Results

A. Feature Selection Using Correlation Matrix:

In our project, we faced the challenge of dealing with numerous features, each potentially contributing to the motion prediction task. However, utilizing all these features could be computationally intensive and might not necessarily improve our model's performance. To address this, as shown in fig. 3, we leveraged the correlation matrix. This matrix provided insights into how each feature correlated with others. By analyzing the correlations, we identified which features were highly interrelated and redundant. Features with high correlation coefficients often conveyed similar information. As a result, we made an informed decision to select a subset of features that were not only relevant but also less correlated with each other. This not only reduced computational complexity but also enhanced our model's interpretive and generalization capabilities. It helped us to maximize prediction accuracy while minimizing complexity.

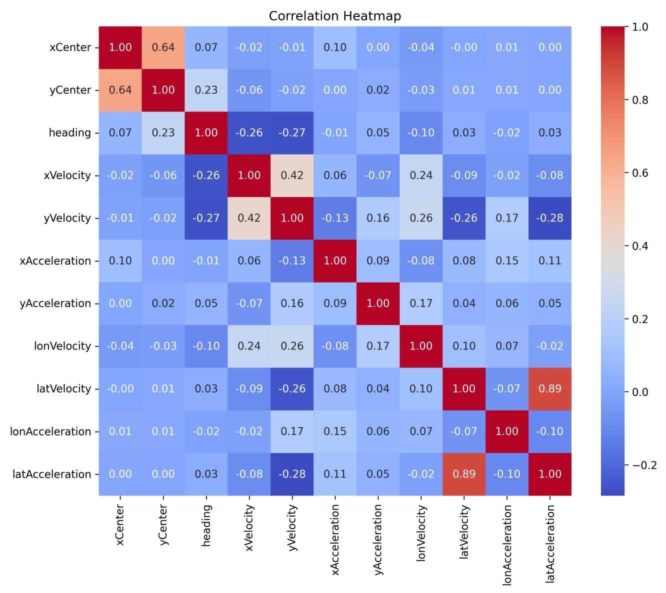


Fig. 4: Correlation Matrix

B. Data Splitting

Data Splitting: We strategically divided our dataset into three subsets: 80-10-10 split. It involves using 80% of the data for training, 10% for validation, and the remaining 10% for testing. Training Set: The foundation of our model's learning process, where it learns from the motion patterns in the data. Validation Set: Used for fine-tuning and adjusting parameters to make right decisions about the architecture of the model Testing Set: This subset allowed us to evaluate real-world performance on unseen data. It is used to see how our model is likely to perform in the real world on unseen data. By carefully partitioning our data, we ensured our machine learning model was well-prepared and rigorously tested, resulting in reliable motion predictions.

C. Hyper-parameters

Grid search is used for hyperparameter optimization find the combination of values that yields the best performance for our machine learning model. Grid search then identifies the set of hyperparameters that resulted in the best performance according to the chosen evaluation metric. Our set of hyper-parameters include: Activation function, Batch size, Learning rate, Number of neurons and Number of Epochs.

Activation Function: Definition: To make our data driven models learn the complex relations between the features, we used different activation functions to introduces non-linearity into the models. It determines the output of a neuron given its weighted sum of inputs. ReLU (Rectified Linear Unit), Sigmoid, Tanh are some of the function that are used during hyperparameter tuning.

Batch Size: The quantity of training samples utilised in a single iteration of training is referred to as the batch size. Faster training may result from larger batch sizes, although this may also need more memory. Although smaller batch numbers increase noise, they occasionally produce more accurate generalisation.

Learning Rate: The step size at which the model's parameters are changed during training is determined by the learning rate. A learning rate that is too high might cause instability, whereas one that is too low can cause the model to get stuck in local minima.

Number of Neurons: It refers to the number of neurons present in the neural network's various layers. The capacity or complexity of the model is determined by the number of neurons. More neurons may be able to pick up more complex patterns, but if they're not well controlled, they may also cause overfitting.

D. Regularization

In machine learning models, we used regularization method for preventing models from fitting the training data too closely, which can result in overfitting. We used different methods such as L1 and L2, Dropout and early stopping. Early stopping and dropout yielded better result in MLP model, whereas, LSTM model yielded better result with early stopping.

L1 (Lasso) and L2 (Ridge) regularization: L1 is mainly used for feature selection. In our data we are dealing with a large number of possibly unimportant features. In this case, L1 regularization introduces sparsity, by encouraging some model coefficients to be exactly zero. L2 is mainly used to shrink the parameters. The magnitude of coefficients is constrained by L2 regularisation without being set to zero.

Dropout: To make out MLP model more adaptive, generalize it from data and avoid overfitting, we used Dropout method. Dropout randomly sets a fraction of neurons outputs to zero during each forward and backward pass. Since the size of our data is not too large, we selected the dropout rate to be 0.2. During each training iteration, 20% of the neurons are randomly ”dropped out” or set to zero.

Early Stopping: As shown in fig. 4, to stop our neural network models from training for an excessive number of epochs, Early Stopping method is used. During training, validation loss is monitored, and the training process is terminated when the metric stops improving or starts degrading. This method is used in MLP as well as LSTM model. Early stopping has helped us optimize the training time. It automatically selects the best number of epochs based on validation performance.

After thorough experimentation, we decided to proceed with early stopping and dropout as it yielded superior results compared to L1 and L2 regularization techniques.

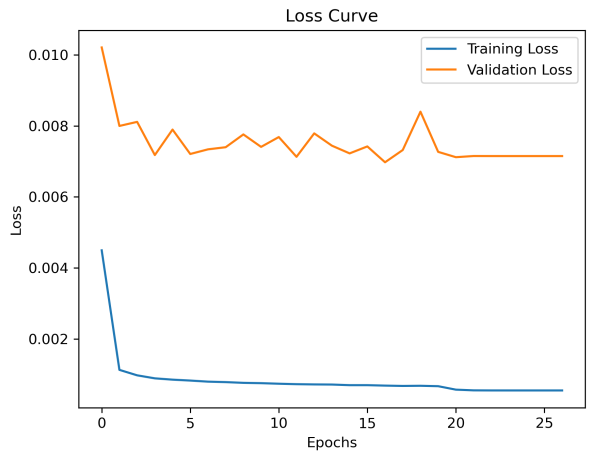


Fig. 5: Loss Curve - Early Stopping

E. Performance Evaluation:

Mean Squared Error (MSE) is calculated by taking the average of the squared differences between actual and predicted values.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

Final Displacement Error (FDE): This error calculates the difference between the predicted final location and the actual final location at the conclusion of the prediction horizon, without taking into account any other time steps in the prediction horizon where a prediction error may occurred.[5]

$$FDE = |\hat{y}_{tf} - \hat{y}_{tf}| \tag{5}$$

Average Displacement Error (ADE): It measures the average difference between the predicted and actual positions of objects over a specific time horizon.

$$ADE = \frac{1}{N}\sum_{i=1}^{N} \|\hat{y}_i - y_i\|_2$$

ADE is used as a final evaluation metric for evaluating entire trajectories of vehicles. We have preffered ADE, as it gives us a detailed understanding of how accurate our predictions are at each individual point in time.

TABLE I: Results - Track ID 25

Models	MSE	ADE	FDE
Bicycle Model(1s)	3.685 m	1.692 m	3.098 m
Constant Acceleration Model(1s)	0.137 m	0.401 m	0.682m
MLP Model (2s)	0.009 m	0.069 m	0.091 m
LSTM Model (2s)	0.006 m	0.046 m	0.063 m
Kalman filter based Constant Velocity model(2s)	$1.105{*}10^{-8}$ m	$3.537{*}10^{-5}$ m	$1.216{*}10^{-5}$ m

F. Results

Physics-based motion models primarily utilize fundamental properties such as dynamic and kinematic characteristics, making them suited for short-term motion predictions, typically spanning durations of less than a second[1]. In contrast, data-driven models, drawing insights from comprehensive datasets, excel in making predictions over longer timeframes, capturing more intricate patterns and dependencies. For Bicycle and Constant Acceleration model we made prediction for 1 second into the future, where as 2 seconds for MLP and LSTM model. After running the models on every track ID and comparing their performance, track ID 25 emerged as the most illustrative choice for presenting our results because of its ability to showcase the strengths of our models.

As shown in Table 1, LSTM model demonstrated commendable performance and yielded very low Average Displacement Error, showcasing its effectiveness in predicting trajectories for car objects. We explored a hybrid approach incorporating Kalman filter and Constant Velocity model which outperformed other methods, achieving the lowest error. However, when compared to the data-driven and hybrid approaches, physics-based models did not perform well and struggled in accurately predicting the trajectories of car objects.

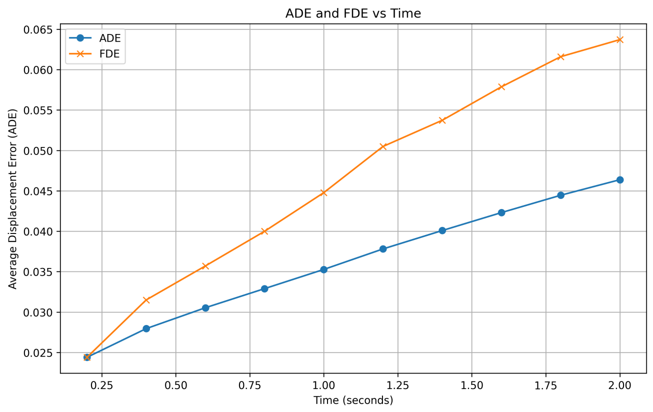


Fig. 6: ADE & FDE: LSTM model

VI. Conclusion and Future Work

The goal of our research was to determine the most accurate model for correctly forecasting the trajectory of car objects in track ID 25, with a particular emphasis on predictions for the next 1 and 2 seconds. According to the results of our study, using machine learning model like the LSTM model to anticipate car movements works incredibly well and results in the lowest Average Displacement Error of 0.046 meter. This shows that as the data driven models are learning from real-world data, they can provide more accurate predictions even in situations where exact physical parameters may not be available. However, without sufficient regularization, data-driven models are prone to overfitting. They could also have trouble with circumstances that drastically deviate from the training data. In case of physics based models, with larger time steps, the models yield larger errors which leads to significant deviations from the true trajectories.

Additionally, to explore an alternative approach, we experimented by combining a Constant Velocity model with a Kalman filter, which yielded very precise results with the Average Displacement Error of $3.5378*10^{-5}$ meter. It has a lot of potential to provide even more accurate predictions about how a vehicle will move using this combination of physics-based and machine learning techniques. While machine learning is excellent for identifying patterns in data, physics-based models perform better when we need predictions for brief periods of time. This demonstrates that both strategies have their advantages, and we can combine them in the future to enhance our motion predictions.

There are a number of intriguing future directions for the study of motion prediction. It may be possible to find a good solution by looking into hybrid models that incorporate components of both data-driven and physics-based techniques, utilising the advantages of both. An exciting topic for future research is the crucial intersection of these techniques. A smooth and adaptable prediction framework might be created by using a switching mechanism that dynamically switches between data-driven and physics-based models depending on the time horizon. This dynamic method would use physics-based models for precise, short-term predictions while making use of the predictive ability of data-driven models for longer-term predictions.

References

[1] Lefèvre et al, "A survey on motion prediction and risk assessment for intelligent vehicles", ROBOMECH Journal 2014.

[2] Paden, Brian et al, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles", IEEE Trans. Intell. Veh. (IEEE Transactions on Intelligent Vehicles), pp. 33–55, 2016.

[3] Bock, Julian et al "The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections", 2020 IEEE Intelligent Vehicles Symposium, pp. 1929–1934,2020.

[4] S. Hiremath and N. Bajcinca, "Machine Learning: Time Series Modeling and RNNs ", Lecture notes, Department of Mechanical and Process Engineering RPTU Kaiserslautern, Germany, summer-semester, 2023.

[5] Mozaffari, Sajjad; Al-Jarrah, Omar Y.; Dianati, Mehrdad; Jennings, Paul; Mouzakitis, Alexandros, "Deep Learning-Based Vehicle Behaviour Prediction for Autonomous Driving Applications: A Review",IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, pp. 33–47, 2022.

[6] S. Hiremath and N. Bajcinca, "Machine Learning: Artificial Neural Networks", Lecture notes, Department of Mechanical and Process Engineering RPTU Kaiserslautern, Germany, summer-semester, 2023.

[7] A. Houenou, P. Bonnifait, V. Cherfaoui and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition", 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, pp. 4363-4369, 2013.

[8] N. Kaempchen, K. Weiss, M. Schaefer and K. C. J. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," IEEE Intelligent Vehicles Symposium, 2004, Parma, Italy, pp. 825-83,2004.

[9] A. Polychronopoulos, M. Tsogas, A. J. Amditis and L. Andreone, "Sensor Fusion for Predicting Vehicles' Path for Collision Avoidance Systems", IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 3, pp. 549-562, Sept.2007.

[10] M. W. Khan, N. Salman, A. Ali, A. M. Khan and A. H. Kemp "A Comparative Study of Target Tracking With Kalman Filter, Extended Kalman Filter and Particle Filter Using Received Signal Strength Measurements", IEEE Transactions on Aerospace and Electronic Systems, vol 59, issue 4, pp. 1–6, August 2023

[11] Prof. Didier Stricker and Dr. Gabriele Bleser, "2D Image Processing: Bayes filter implementation: Kalman filter", Lecture notes, Department of Augmented Vision, RPTU Kaiserslautern, Germany, summer-semester, 2023.