Predictive Analysis of Vehicular Lane Changes: An Integrated LSTM Approach

Hongjie Liu 1,2, Keshu Wu 3, Sicheng Fu 3, Haotian Shi 3,4,* and Hongzhe Xu 1

1 Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an 710049, China; hj_popel@stu.xjtu.edu.cn (H.L.); xuhz@mail.xjtu.edu.cn (H.X.)
2 Guangdong Planning and Designing Institute of Telecommunications Co., Ltd., Guangzhou 510630, China
3 Civil and Environmental Engineering, University of Wisconsin, Madison, WI 53705, USA; kwu84@wisc.edu (K.W.); sfu42@wisc.edu (S.F.)
4 College of Traffic and Transportation Engineering, Tongji University, Shanghai 201804, China
* Correspondence: bshi84@wisc.edu; Tel.: +1-(608)-5718009

Abstract: In the rapidly advancing domain of vehicular traffic management and autonomous driving, accurate lane change predictions are paramount for ensuring safety and optimizing traffic flow. This study introduces a comprehensive two-stage prediction model that harnesses the capabilities of long short-term memory (LSTM) for anticipating vehicular lane changes. Initially, we employed a variety of models, such as regression methods, SVMs, and a multilayer perceptron, to categorize lane change behaviors. The dataset was then segmented based on vehicle trajectories and lane change patterns. In the subsequent phase, we utilized the superior classification outcomes from LinearSVC to curate our training data. We developed two dedicated LSTM networks tailored to specific datasets: the lane-keeping LSTM (LK-LSTM) and the lane-changing LSTM (LC-LSTM). By integrating insights from both models, we achieved a comprehensive prediction of vehicular lane changes. Our results indicate that the unified prediction model markedly enhances prediction precision. Accurate lane change predictions offer valuable contributions to advanced driver-assistance systems (ADAS), with the potential to minimize traffic mishaps and enhance traffic fluidity. As we transition to a more autonomous automotive era, refining these predictions becomes essential in seamlessly merging human and automated driving experiences.

Keywords: LSTM; lane change prediction; LinearSVC

1. Introduction

In today’s traffic environment, marked by its high density, lane changes have become crucial maneuvers at the microscopic level. For automated vehicles, anticipating the immediate driving state is essential for informed decision making. Accurate foresight into lane changes can significantly reduce crash risks. Moreover, it can also boost traffic efficiency and improve driving comfort by eliminating unnecessary maneuvers, thereby directly enhancing traffic safety. To enhance urban traffic safety and efficiency, this study aims to develop a precise and efficient model for classifying vehicle lane change maneuvers and predicting trajectories. This objective will be achieved through a comprehensive analysis of traffic conditions, an in-depth review of the relevant literature, and the application of advanced artificial intelligence modeling techniques, ultimately refining vehicle behavior prediction.

techniques. The model leverages both a deep belief network (DBN) and long short-term memory (LSTM) neural networks to represent the LC process, which encompasses lane change decisions (LCDs) and lane change implementation (LCI). Benterki et al. (2019) [4] concentrated on forecasting lane changes by analyzing surrounding vehicles on highways, harnessing both the SVM and Bayesian filter methodologies. Su (2018) [5] introduced an LSTM model that integrated historical vehicle trajectories with the present state of neighboring vehicles to predict future trajectories. Liu Y (2019) [6] presented a decision-making model for autonomous lane changes, drawing on the principles of benefit, safety, and tolerance. This model was crafted through an in-depth analysis of the factors influencing lane transitions in autonomous vehicles. Mahajan V. (2020) [7] introduced an end-to-end machine learning model designed to predict lane change maneuvers using a minimal set of features from unlabeled data. K. Shi (2022) [8] developed a neural network architecture that integrates the attention mechanism, bidirectional long-short-term memory (BiLSTM), and a temporal convolutional neural network (TCN) to emulate and forecast joint behaviors. Several researchers have employed diverse methodologies to predict vehicle lane changes, viewing the problem from multiple angles and achieving noteworthy results. Additionally, researchers have attempted to improve the accuracy of vehicle lane-changing predictions through hybrid modeling approaches, as demonstrated by Zhang et al. (2019) [9,10] and Shi et al. (2022) [8]. These efforts have resulted in a notable enhancement in predictive accuracy. In addition to this, lane change prediction in autonomous driving is a burgeoning area of research. Yi Zhang proposed a framework that focuses on feature learning for lane change prediction. The objective is to gain a deep and comprehensive understanding of lane change behaviors while achieving high performance based on the chosen features [11]. Similarly, Yuan Shen introduced AutoPreview, a framework that allows consumers to preview an autopilot system’s potential actions in real-world driving conditions before they are deployed [12].

Despite the notable accomplishments of the aforementioned research, a majority failed to segment data based on distinct behavioral patterns during lane changes. Additionally, their reliance on a singular prediction model restricts adaptability across varied scenarios. Addressing these limitations, our study introduces a nuanced two-stage model that synergizes data situation classification with vehicle lane change prediction models, with each tailored for specific scenarios. In our work, we harness data from the Next Generation Simulation (NGSIM) program [13], a renowned vehicle trajectory dataset from I-80 in Emeryville, CA. Our trajectory prediction model builds upon a three-phase prediction process:

1. Lane change labeling and prediction: Vehicles’ lane-changing maneuvers in upcoming timesteps are labeled. We then employ various regression models to both train the lane change model and forecast the lane-changing behavior. During this phase, each timestep of the vehicle movement trajectory is classified for lane-changing maneuvers. For example, a zero denotes a lane-keeping maneuver, while a one signifies a lane-changing maneuver. Several algorithms, including regression methods, SVMs, and a multilayer perceptron, will be utilized for this classification. A continuous series of zeros or ones will then depict the maneuvers for that journey.

2. Trajectory prediction based on maneuvers: Each categorized maneuver segment undergoes a distinct machine learning trajectory prediction model application. This entails the lane-keeping prediction model processing data exclusive to lane keeping, while the lane-changing prediction model exclusively processes lane-changing data. Our model of choice for this task is the long short-term memory (LSTM) algorithm, a subset of recurrent neural networks. We theorize that employing maneuver-specific models could enhance the overall predictive accuracy.

3. Integration model: The models developed in steps 1 and 2 are integrated into a new model. The maneuver classification model takes precedence to predict the vehicle’s driving behavior (either a lane change or lane keeping). Depending on the predicted maneuver for each timestep, an appropriate trajectory prediction model is applied. This method facilitates a holistic prediction of the vehicle trajectory.
The key contributions of this research are as follows:

1. An empirical analysis is conducted on the performance of various classification models using the NGSIM dataset. The results reveal that logistic regression and LinearSVC outperformed other models in terms of classification efficacy.

2. Regarding vehicle behavior prediction, segmenting the data beforehand and tailoring prediction methods based on the classification outcomes renders the model more versatile. Consequently, this approach achieves a heightened prediction accuracy. A visual representation of our methodology is provided in Figure 1.

![Method Workflow Diagram]

Figure 1. Method workflow.

This paper is structured as follows. Section 2.1 gives an introduction to the NGSIM data, detailing their preprocessing and chosen features. Section 2.2 has a deep dive into our algorithmic approach and subsequent predictions. Section 3 shows detailed results from both the lane change and trajectory prediction models. Section 4 contains insightful analysis of our results, culminating in a discussion on future research directions.

2. Methodologies

This section delves into the foundational aspects of our research. We commence with an introduction to the NGSIM dataset, detailing its preprocessing and the features we deemed essential for our study. This dataset forms the bedrock upon which our subsequent analyses are built. Following the data introduction, we embark on a deep dive into our algorithmic approach, elucidating the methodologies employed for both maneuver classification and trajectory prediction. A visual representation of our methodology, depicted in Figure 1, aids in understanding the workflow and interplay between various stages of our research. The methodology comprises two computational models: the maneuver...
classification model and the trajectory prediction model. The input parameters and output
decision variables for each model are detailed in Tables 1 and 2.

Table 1. Maneuver classification parameters.

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>the timestamp</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>the vehicle ID</td>
</tr>
<tr>
<td>$\delta$</td>
<td>the lane ID</td>
</tr>
<tr>
<td>$v$</td>
<td>the vehicle speed</td>
</tr>
<tr>
<td>$v_x$</td>
<td>the lateral speed</td>
</tr>
<tr>
<td>$\theta$</td>
<td>the steering angle</td>
</tr>
<tr>
<td>$a$</td>
<td>the vehicle acceleration</td>
</tr>
<tr>
<td>$x$</td>
<td>the lateral position</td>
</tr>
<tr>
<td>$y$</td>
<td>the longitudinal position</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decision variable</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>the lane-changing maneuver ($-1/1$)</td>
</tr>
</tbody>
</table>

Table 2. Trajectory prediction parameters.

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>the timestamp</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>the vehicle ID</td>
</tr>
<tr>
<td>$\delta$</td>
<td>the lane ID</td>
</tr>
<tr>
<td>$v$</td>
<td>the vehicle speed</td>
</tr>
<tr>
<td>$v_x$</td>
<td>the lateral speed</td>
</tr>
<tr>
<td>$\theta$</td>
<td>the steering angle</td>
</tr>
<tr>
<td>$a$</td>
<td>the vehicle acceleration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decision variable</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>the lateral position</td>
</tr>
<tr>
<td>$y$</td>
<td>the longitudinal position</td>
</tr>
</tbody>
</table>

2.1. Data Extraction and Feature Engineering

To derive meaningful insights, the availability and processing of appropriate data
are indispensable. Here, we delve into the nature and structure of our dataset. The initial
dataset comprises approximately 11.8 million rows and 25 columns. Each row provides
data pertaining to a single vehicle at a specific time step, with these steps spaced at 0.1 s
intervals. The 25 feature columns can be categorized as follows:

- Timestamp information (i.e., Frame_ID, Total_Frames, Global_Time);
- Vehicle position (i.e., Local_X, Local_Y, Global_X, Global_Y, Lane_ID);
- Vehicle basic information (i.e., v_length, v_Width, v_Class);
- Motion information (i.e., v_Vel, v_Acc).

The following subsections give the details of the methodology.

2.1.1. Data Extraction

While the original dataset is expansive, not every piece of data is pertinent to our
primary focus on vehicle motion. Thus, it becomes imperative to identify and retain only
the most crucial features.

The original dataset encompasses 25 features. However, features unrelated to vehicular
motion, such as basic vehicle information and information about surrounding vehicles
not associated with movement, should be omitted. Within the timestamp category, only a
singular feature is required to pinpoint the exact moment of the motion; hence, Global_Time
is retained. For denoting the precise vehicle position, features such as Local_X, Local_Y, and
Lane_ID are vital.
To summarize, out of the 25 feature columns, we distilled the dataset to a set of 7 pivotal features that encapsulate the essence of vehicular movement. These are the timestamp \((t)\), vehicle ID \((\sigma)\), lane ID \((\delta)\), speed \((v)\), acceleration \((a)\), longitudinal position \((y)\), and lateral position \((x)\).

Given the vast amount of data rows, we will focus solely on data for vehicles with an ID less than 1500 and those that have executed a lane change. This refined dataset comprises 54,463 rows and 7 columns.

2.1.2. Maneuver Labeling

After filtering out the significant data, understanding the behavior of vehicles during lane-changing maneuvers is the next step. This involves further data categorization and mathematical modeling.

To more aptly represent the intricacies of lane-changing maneuvers, it is imperative to introduce two additional features related to lateral motion: lateral speed \((v_x)\) and steering angle \((\theta)\).

The steering angle is defined as the angle between the vehicle’s front direction and its steering direction. It can be inferred from the ratio of the lateral position difference to the longitudinal position difference. The computation for the steering angle can be performed using Equation (1):

\[
\theta = \arctan \frac{\Delta x}{\Delta y} = \arctan \frac{x_t - x_{t-1}}{y_t - y_{t-1}}
\]  

(1)

Given that the speed \(v\) represents the longitudinal speed, the lateral speed can be deduced from the steering angle in Equation (2):

\[
v_x = v \cdot \tan \theta
\]  

(2)

In the context of vehicular movement, lane-changing maneuvers are specifically defined by the vehicle’s intention to transition from one lane to another. Each action in a vehicle’s trajectory sequence can be taxonomically categorized as either lane keeping, denoted by \(-1\), or lane changing, denoted by \(+1\). For those vehicles that engage in a lane-changing operation, the trajectory data can be parsed into three discrete phases: (1) lane keeping before the lane change; (2) a lane-changing event; and (3) lane keeping after the lane change. Subsequently, a continuous sequence of trajectory data typically manifests as an alternating pattern of these categorized states, represented by successive runs of \(+1\) and \(-1\), which reflect the oscillatory nature of lane keeping and lane changing during a journey.

Analytically, the behavior of the lateral speed \(v_x\) and steering angle \(\theta\) serve as instrumental indicators for identifying the phases of a lane change. In the lane-keeping phase, both \(v_x\) and \(\theta\) are approximately zero, signifying static lateral motion. Conversely, during the lane-changing event, both variables exhibit a discernible pattern, typically characterized by an initial increase followed by a subsequent decrease. This characteristic behavior of \(v_x\) and \(\theta\) thus provides a reliable metric for distinguishing lane-changing maneuvers from lane-keeping actions.

To systematically analyze vehicular maneuvers, we initially subjected the raw data of the lateral speed \(v_x\) and steering angle \(\theta\) to a smoothing operation. Subsequently, we computed their respective mean values. At each time step, a vehicle was identified as undergoing a lane-changing event if both \(v_x\) and \(\theta\) concurrently exceeded their respective mean values. In such cases, the maneuver was categorically labeled as a lane-changing event and denoted by \(+1\). Otherwise, the vehicle was considered to maintain its current lane, and the maneuver was tagged as lane keeping, denoted by \(-1\).

Figure 2 provides several examples of vehicular lane change trajectories. In this representation, red points are employed to denote lane change maneuvers and are annotated with a label of \(+1\). Conversely, blue points are used to signify lane-keeping activities before or after lane changing and are marked with a label of \(-1\).
2.1.3. Feature Engineering

For the maneuver classification segment, the input features for every vehicle at each time step are detailed as follows:

- Vehicle ID \( \sigma \);
- Speed \( v \);
- Lateral speed \( v_x \);
- Steering angle \( \theta \);
- Lateral position \( x \)

The corresponding output label represents the lane-changing maneuver, denoted by \( m \), mathematically as shown in Equation (3):

\[
X = [t, id, \delta, v, v_x, \theta, a, x, y] \quad \text{and} \quad Y = [m]
\]  

(3)

Moving on to the trajectory prediction segment, the employment of deep learning algorithms inevitably entails increased computational overhead. Following the classification of vehicular activities, our analytical emphasis shifted toward identifying the attributes that exerted the most pronounced influence on these activities. Through meticulous investigation of feature correlations and critical evaluation of attribute significance, we were
positioned to refine our dataset, retaining only those features that were of paramount importance for our analytical objectives.

To streamline the input data for model training, it is essential to retain only the principal features. Figure 3 presents a correlation matrix illustrating the correlation coefficients among the features. As evidenced by the matrix, there is a strong correlation between the time feature \( t \) and the feature \( y \). Similarly, \( \sigma \) is highly correlated with \( x \). Given these significant correlations, we suggest omitting the features \( t \) and \( \delta \) to decrease data dimensionality.

**Figure 3.** Correlation matrix of the features.

Furthermore, it is imperative to focus on the primary features pertinent to the labels, implying that the most influential features should be chosen. To evaluate the significance of each feature and illustrate its correlation with the output label, we generated a bar chart, shown in Figure 4, using the `sklearn.feature_selection.SelectKBest` function. This chart clearly indicates that features \( \sigma, v, v_x, \theta \), and \( x \) registered higher scores that contributed the most to the variable, underscoring their importance. As such, these features are the ideal candidates for selection.

**Figure 4.** Scores of the features.

Thus, the input features for every vehicle at each respective time step are as follows:

- Timestamp \( t \);
- Vehicle ID \( \sigma \);
- Lane ID \( \delta \);
- Speed \( v \);
- Lateral speed \( v_x \);
- Steering angle \( \theta \);
- Acceleration \( a \).
The outputs are the longitudinal position $y$ and lateral position $x$, shown mathematically in Equation (4):

$$X = [t, id, \delta, v, v_x, \theta, a] \quad \text{and} \quad Y = [x, y] \quad (4)$$

2.1.4. Time Series Input and Output

To reinforce the importance of the time series data, we reiterate the need to consider past data when making future predictions. While this has similarities with the previous section, it is paramount to understand the significance of including past movements for an accurate model.

Vehicular movements over time form a sequential, interconnected pattern, much like any time series data. Capturing this continuous flow and understanding how past states affect future outcomes is pivotal for our analysis.

Vehicle motion is inherently continuous, implying that past movements can influence future ones. Predicting maneuvers and trajectories thus becomes a time series forecasting challenge. For each time step, it is important to incorporate data from preceding time steps into the input features. Moreover, to enhance the model’s efficacy, considering maneuvers from a prior duration as a distinct feature could be beneficial.

Suppose that the current time step is $T$, and the number of the previous time step is $T_p$. To generate a continuous sequence of previous states, the reframed input $X_T$ and output $Y_T$ at time step $T$ can be expressed as in Equation (5):

$$X_T = [x_{T-1}, x_{T-2}, \ldots, x_{T-T_p}], Y_T = [y_T] \quad (5)$$

For the classification models, the state at the time step $k$ can be represented as in Equation (6):

$$x_k = [t_k, \sigma, \delta_k, v_k, (v_k)_k, \theta_k, a_k, y_k, x_k, m_k], k \in (T - T_p, T) \quad (6)$$

The matrix $X_T$ is flattened to a $1 \times [n_{features} \times n_{T_p}]$ vector. The dimension of the input data $X$ is $n_{rows} \times [n_{features} \times n_{T_p}]$. The output can be expressed as $Y_T = [m_T]$. The dimension of the output data $Y$ is $(n_{rows} \times 1)$.

Similarly, for the trajectory prediction model, the state at the time step $k$ can be represented as in Equation (7):

$$x_k = [t_k, \sigma_k, \delta_k, v_k, (v_k)_k, \theta_k, a_k, x_k, y_k], k \in (T - T_p, T) \quad (7)$$

The matrix $X_T$ is directly used as the input for each time step. The input data $X$ is a tensor with the dimension of $n_{rows} \times n_{features} \times n_{T_p}$. The output can be expressed as $Y_T = [x_T, y_T]$. The dimension of the output data $Y$ is $(n_{rows} \times 2)$.

When the number of previous time steps $T_p$ is large, the dimensionality of $X_T$ significantly increases. This can lead to a slower training process and might adversely impact the model’s performance.

In conclusion, for the maneuver classification problem, considering both the correlation matrix and score of the features, the selected features are $v, v_x, \theta, a$, and $x$. The input is expressed as $X_T = [x_{T-1}, x_{T-2}, \ldots, x_{T-T_p}]$. For a specific time step $k$, $x_k = [v_k, (v_k)_k, \theta_k, x_k, m_k]$. The output is $Y_T = [m_T]$.

2.1.5. Training, Validation, and Testing

Finally, after data extraction, labeling, and feature selection, the crux of our study lies in the application and evaluation of machine learning models. Using the majority of our data to train these models and the remainder to test them ensured we had a reliable, robust solution to our problem.

Of the available data, 70% were allocated to the training set, while the remaining 30% were designated for the testing set. The optimal model underwent K-fold cross-validation, where the training dataset was divided into k distinct subsamples. In this procedure,
k − 1 folds [14] of the dataset served as the training data, while the remaining fold acted as the validation set. This cycle was repeated k times. After these iterations, the model demonstrating the highest validation accuracy was selected for the testing phase.

2.2. Algorithms and Program

In this subsection, we elucidate the algorithmic approach underpinning our research. Building on the foundational data introduced in the previous section, we detail the methodologies employed for maneuver classification and trajectory prediction. The workflow, as visually represented in Figure 1, showcases the interplay between various stages of our research, offering readers a clear understanding of how data are processed, analyzed, and transformed into meaningful insights.

2.2.1. Maneuver Classification

Regression

The regression model consists of multiple methods that can predict the target variable (y) based on one or multiple features (x). Mathematically, we can define the regression model as a function of the label y and feature vector x.

In this experiment, we use four kinds of regression methods—logistic regression [15], stochastic gradient descent (SGD) regression [16], ridge regression [17], and lasso regression [18]—to predict the lane change maneuver based on the input features. The Sk-learn package will be used to build the models.

Logistic regression measures the relationship between the dependent variable (the label to be predicted) and one or more independent variables (features) by using its inherent logistic function (sigmoid function) to estimate the probability.

The linear regression algorithm aims to create a polynomial function to predict the label value of a new sample. In the normal gradient descent process, the cost of all samples in the data set is calculated for each iteration, and the value of the parameter θ is adjusted accordingly to make the cost function J(θ) gradually approach the most merit (lowest point). The stochastic gradient descent (SGD) method randomly selects a sample each time to adjust the parameters. Compared with the default gradient descent (using all samples to calculate the parameters), stochastic gradient descent solves the computation demand problem caused by the excessive number of samples.

Ridge regression is a biased estimation regression method dedicated to collinearity data analysis and is essentially an improved least squares estimation method. Ridge regression abandons the least square method’s unbiased nature and obtains more realistic and reliable regression coefficients at the cost of losing some information and reducing accuracy. Its fitting to ill-conditioned data is stronger than the least squares method.

Like ridge regression, the least absolute shrinkage and selection operator also adds a penalty to the regression coefficient’s absolute value. Aside from that, it can reduce bias and improve the accuracy of linear regression models. Lasso regression uses absolute values instead of square values in the penalty part. This leads to a penalty (that is, it constrains the sum of the estimates’ absolute values) to make some parameter estimates equal to zero.

Having explored regression techniques, another powerful method comes to light which specializes in binary classification tasks.

Support Vector Machine (SVM)

The support vector machine (SVM) [19] is a perfect algorithm for binary classification problems. The data can be mapped to higher dimensions and be separated by a hyperplane. Since the input data might be nonlinearly separable, both linear support vector classification models and support vector classification models using nonlinear kernels will be used. To design a better SVM model, the performance of the models using different kernel functions will be compared. The kernel functions are as follows:

- Polynomial: \((x^Ty + 1)^p\);
- Radial basis function: \(\exp\left(-\frac{1}{2\sigma^2}\|x - y\|^2\right)\);
Sigmoid: \( \text{tanh}(\beta_0 x^T y + 1) \).

While SVMs provide a distinct approach to binary classification, neural networks present another versatile avenue for maneuver classification, especially for complex, non-linear data.

**Multilayer Perceptron (MLP)**

The multilayer perceptron [20] is a class of feedforward artificial neural networks. An MLP utilizes a supervised learning technique of backpropagation for training. Its multiple layers and nonlinear activation function can distinguish data that are not linearly separable.

An MLP in the experiment is built with one input layer with the dimensions of the input training data, 2 hidden layers with 100 neurons and an ReLU activation function, and 1 output layer with a tanh activation function. The structure of the network is shown in Figure 5. In this diagram, the yellow rectangles denote the input data featured in the input layer, while the blue circles symbolize the neurons residing in the hidden layer. Additionally, the green circles represent the activation functions utilized in the output layer.

**Figure 5. The structure of the MLP.**

### 2.2.2. Trajectory Prediction

A long short-term memory (LSTM) [21] network is a specialized form of the recurrent neural network (RNN) [22] which is uniquely equipped to address the vanishing gradient issues typical of traditional RNNs. Central to its design is a memory cell, facilitating the retention of information over extended periods. Guided by its input, output, and forget gates, the network manages the intake, dispensing, and disposal of information. Such a configuration is adept at capturing long-term dependencies, a trait evidenced by its efficacy in vehicle trajectory predictions as documented by Su (2018) [5].

For model construction, we will leverage the Keras package. Given that the input \( X \) is a 3D tensor, the neurons in the input layer adopt a shape of \((n_{\text{features}} \times n_{TP})\). The architecture incorporates an LSTM layer with 256 neurons followed by a dropout layer to mitigate overfitting, collectively serving as the hidden layers. The output layer comprises two neurons, signifying \( y \) and \( x \). Optimization is carried out using the Adam optimizer, while the chosen loss function is the mean squared error. The network undergoes training for 100 epochs.

Initially, a holistic LSTM model—termed Simple-LSTM—is trained using the entire dataset, encompassing both lane-keeping and lane-changing scenarios. Subsequently, we delineate two specialized LSTM models: one focused on lane keeping (LK-LSTM) and the other focused on lane changing (LC-LSTM). The dataset is bifurcated accordingly; one subset is replete with lane-keeping data, while the other is exclusively constituted of lane-changing data. Each segment is then independently harnessed to train its respective model.

Having established the methods for predicting vehicle trajectories, the natural progression is to integrate these predictions into a holistic framework, ensuring that each predicted maneuver seamlessly aligns with the subsequent vehicular position.
2.2.3. Integrated Prediction Model

We developed a comprehensive prediction model. For each vehicle, its trajectory is reconstructed utilizing this model. At every timestep, the model initially forecasts the lane change maneuver. If the predicted maneuver yields a value of zero, then the LK-LSTM model is engaged to predict positions for the subsequent timestep. Conversely, if the maneuver does not equate to zero, then the LC-LSTM model is employed for prediction. To gauge the accuracy of the predicted trajectory, it is compared against the true trajectory using the root mean squared error as a metric.

Regarding the algorithm’s practical applications, it should be noted that the learning datasets employed in our study are derived from the ground truth NGSIM field datasets. It captures a broad spectrum of driving behaviors like acceleration, deceleration, and uniform speed across various scenarios. Despite utilizing offline data, our integrated approach is adept at learning from these ground truth behaviors, enabling the model to generate realistic trajectory predictions. In practical applications, our model is engineered to operate in real time. It takes multiple timesteps of historical trajectory data as input and outputs multiple timestep prediction trajectories at each timestep. Leveraging advancements in vehicle localization and sensing technologies, the model can collect real-time data on historical trajectories as input. Furthermore, the computational time required for each timestep is minimal—far less than 0.1 s—making it feasible for real-time implementation.

3. Results and Analysis

Delving into our analysis, we will initially focus on the methods of maneuver classification.

3.1. Maneuver Classification

3.1.1. Regression

Understanding the relationship between dependent and independent variables is of paramount importance for the objectives of this study. To this end, we initiated a regression experiment as our foundational analytical approach. In the preparatory phase, all features in the training dataset were normalized and subsequently fed into four distinct regression models for training. Upon the completion of this training phase, immediate evaluation was conducted using a separate testing dataset. We then furnished both training and testing accuracies and engaged in a comparative analysis to evaluate the relative performance of the four models.

To narrow down our focus, we proceeded with a more targeted training process using a carefully selected set of pivotal features, specifically \([v, vx, theta, pos_x, maneuver]\). Based on the performance metrics, the model demonstrating the highest efficacy was earmarked for more in-depth evaluation. It is worth noting that logistic regression and stochastic gradient descent (SGD) regression emerged as strong contenders, displaying comparable performance and thus becoming viable candidates for further scrutiny.

Following the initial regression tests, we dived deeper to ascertain the results. Table 3 elucidates the training and testing accuracies associated with each model. A noteworthy observation is that logistic regression emerged as the most efficacious model, albeit with marginally higher performance than SGD regression. Furthermore, the performance of logistic regression remained consistently superior. Ridge and lasso regression, although displaying commendable accuracy (attaining 97.54% on the testing dataset), were not as competitive when juxtaposed with the logistic and SGD regression models.
Table 3. Accuracy of regression models.

<table>
<thead>
<tr>
<th>Models</th>
<th>All Features</th>
<th>Selected Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (%)</td>
<td>Testing (%)</td>
</tr>
<tr>
<td>SGD Regression</td>
<td>99.59</td>
<td>99.59</td>
</tr>
<tr>
<td>Ridge Regression</td>
<td>97.60</td>
<td>97.54</td>
</tr>
<tr>
<td>Lasso Regression</td>
<td>97.60</td>
<td>97.54</td>
</tr>
<tr>
<td>LinearSVC</td>
<td>99.18</td>
<td>99.17</td>
</tr>
</tbody>
</table>

Recognizing the indispensability of comprehensive evaluation protocols, we took meticulous steps to further validate our findings. To bolster the robustness of our analysis, we employed K-fold cross-validation, a technique widely acknowledged for mitigating overfitting and enhancing model performance. Specifically, for logistic regression, the dataset was partitioned into eight subsets, and K-fold cross-validation was conducted. Table 4 presents the confusion matrices for both the training and validation sets corresponding to the top-performing regression model. The results revealed an encouraging trend: While the peak validation accuracy reached an impressive 99.75%, the testing accuracy of the most optimal model remained stable at 99.63%.

Table 4. Confusion matrices of the top-performing regression model.

<table>
<thead>
<tr>
<th>Best Training Set</th>
<th>Predicted Labels</th>
<th>Best Validation Set</th>
<th>Predicted Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>−1</td>
<td>+1</td>
<td>−1</td>
</tr>
<tr>
<td>True Labels</td>
<td>38,611</td>
<td>93</td>
<td>True Labels</td>
</tr>
<tr>
<td></td>
<td>93</td>
<td>8855</td>
<td>+1</td>
</tr>
</tbody>
</table>

Having assessed the capabilities of the regression models, we then pivoted our investigatory focus toward another powerful computational technique—support vector machines—to broaden the analytical spectrum of our study.

3.1.2. Support Vector Machine (SVM)

When initiating our investigation into support vector machines (SVMs), we commenced with a comprehensive training phase that incorporated all available features. These were input into each of four distinct SVM models for evaluation. The subsequent testing phase relied on a separate dataset to determine the test accuracy for each model. A model registering high values for both training and testing accuracies was subsequently considered superior in performance.

In the next phase, training was narrowed to incorporate only those features deemed most significant. The performance metrics—specifically the training and testing accuracies—were utilized as yardsticks to gauge the efficacy of the models, akin to the approach in the preliminary phase.

An in-depth analysis of these preliminary SVM experiments ensued, with Table 5 elucidating the training and testing accuracies for each model. Remarkably, the discrepancies between the training and testing accuracies across models were minimal, suggesting consistent performance. The support vector classification (SVC) model with a sigmoid kernel was found to underperform, yielding the least accurate predictions. Models using both polynomial and radial basis function (RBF) kernels secured accuracies around 81% when trained on all features. However, this accuracy escalated dramatically to nearly 98% following feature optimization. The linear SVC model stood out, maintaining consistently high accuracies throughout both experimental phases.
Table 5. Accuracy of SVM models.

<table>
<thead>
<tr>
<th>Models</th>
<th>All Features</th>
<th>Selected Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (%)</td>
<td>Testing (%)</td>
</tr>
<tr>
<td>SVC (kernel = poly)</td>
<td>81.21</td>
<td>81.86</td>
</tr>
<tr>
<td>SVC (kernel = rbf)</td>
<td>81.21</td>
<td>81.86</td>
</tr>
<tr>
<td>SVC (kernel = sigmoid)</td>
<td>67.72</td>
<td>29.48</td>
</tr>
</tbody>
</table>

Given its exceptional performance, the linear SVC model was identified as the best-performing model and therefore earmarked for more detailed evaluations in subsequent phases of the study. As we transitioned to a deeper evaluation of the linear SVC model, we employed K-fold cross-validation, a universally endorsed methodology for mitigating overfitting and enhancing model generalization. Specifically, the data were partitioned into eight subsets. Each subset alternately served as the validation set, while the remaining subsets formed the training dataset. The model that achieved the highest validation accuracy was deemed optimal for the study.

Table 6 presents the confusion matrices for both the training and validation sets corresponding to the top-performing SVM model. Across the eightfold validations, the peak training accuracy reached 99.61%, and the peak validation accuracy climbed to 99.75%. Importantly, these figures were closely aligned with the consistent 99.63% accuracy observed when the model was trained and tested on the full dataset.

Table 6. Confusion matrices of the top-performing SVM model.

<table>
<thead>
<tr>
<th>Best Training Set</th>
<th>Predicted Labels</th>
<th>Best Validation Set</th>
<th>Predicted Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>−1</td>
<td>+1</td>
<td>−1</td>
</tr>
<tr>
<td>True Labels</td>
<td>38,612</td>
<td>92</td>
<td>5517</td>
</tr>
<tr>
<td>+1</td>
<td>92</td>
<td>8855</td>
<td>8</td>
</tr>
</tbody>
</table>

Concluding our comprehensive SVM analysis, we shifted our attention to exploring the potential of multilayer perceptrons as another instrumental tool in our research arsenal.

3.1.3. Multilayer Perceptron (MLP)

For commencing our analysis with the multilayer perceptron (MLP) methodology, the first step entailed the application of min-max normalization to the data. This preprocessing was essential for the construction of the MLP model into which all available features were initially fed. The model was refined using binary cross-entropy as the loss function and parameterized with an epoch size of 50, a batch size of 128, and a validation split constituting 20% of the dataset. With these preliminary configurations in place, training of the Keras-based MLP model proceeded.

To enhance the model focus, a subsequent phase employed only a select set of heavily weighted features, namely \([id, v, v_x, \theta, x]\), for training, adhering to the same procedural steps outlined in the initial phase.

Post-experimentation, we undertook an in-depth exploration of the resulting data. Table 7 provides details on both the training and testing accuracies for the MLP model.

Table 7. Accuracy of MLP model.

<table>
<thead>
<tr>
<th>All Features</th>
<th>Selected Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (%)</td>
<td>Testing (%)</td>
</tr>
<tr>
<td>98.3</td>
<td>98.33</td>
</tr>
</tbody>
</table>
To facilitate a robust and critical assessment of these outcomes, an eightfold cross-validation method was applied. Table 8 presents the confusion matrices for both the training and validation sets corresponding to the top-performing MLP model. Across the multiple validation runs, the top-performing training and validation accuracies were found to be 99.61% and 99.75%, respectively, with an average validation accuracy stabilizing at 99.63%. Intriguingly, when the most optimally validated model was subjected to final testing, it exhibited an accuracy rate of 91.2%.

Table 8. Confusion matrices of the top-performing MLP model.

<table>
<thead>
<tr>
<th></th>
<th>Best Training Set</th>
<th>Best Validation Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted Labels</td>
<td>Predicted Labels</td>
</tr>
<tr>
<td></td>
<td>−1  +1</td>
<td>−1  +1</td>
</tr>
<tr>
<td>True Labels −1</td>
<td>38,610 92</td>
<td>True −1</td>
</tr>
<tr>
<td></td>
<td>+1  92 8855 labels</td>
<td>+1  8 1273</td>
</tr>
</tbody>
</table>

In the segment concerning maneuver classification, nearly all employed methodologies—except for the SVM model using a sigmoid kernel—yielded highly commendable predictive accuracies, the majority of which exceeded 98%. Additionally, training the models on the selected features yielded notably improved prediction accuracies when compared with utilizing the complete feature set.

Across all techniques deployed for lane change maneuver prediction, both logistic regression and LinearSVC emerged as top performers. Each registered an impressive accuracy of 99.63% on the testing dataset when trained on the selected features. This level of accuracy remained consistently high, even when subjected to eightfold cross-validation.

To evaluate the model’s predictive capacity for individual vehicle trajectories, we compared the ground truth maneuvers to those predicted by the model at each time step. Figure 6 graphically represents this comparison of three vehicles. While the blue line delineates the true maneuvers, the red line plots the predicted maneuvers. The black cross mark indicates the misclassified predicted maneuver. Although a close congruence between the predicted and actual data points is readily observable, there are some deviations, particularly at the beginning and the end of the lane-changing sequences. Specifically, at the first time step of the lane-changing process, the model tended to inaccurately forecast a lane-keeping action (denoted by zero) in instances where a lane change action (designated by one) actually occurred. At the first time step after the lane-changing process, the model forecasted a lane change action instead of a lane-keeping action. This constituted the principal source of predictive inaccuracies across the multiple vehicles under study.

In summary, our maneuver classification model demonstrated strong performance, albeit with room for improvement at specific lane change junctures.

3.2. Trajectory Prediction

We now pivot to another crucial aspect: the trajectory prediction. Before model training, both the training and testing sets underwent normalization. Each vehicle’s data were transformed into a sequential time series. Subsequently, sequences from all vehicles were amalgamated into a unified set.

During the training process, the testing set doubles as the validation set. For every epoch, losses for both the training and validation sets are computed. As depicted in Figure 7, there was a noticeable downward trend in the losses. After 100 epochs, the losses stabilized at an acceptable threshold.

The root mean square error (RMSE) was computed for both the training and testing sets for their respective outputs. Table 9 presents the results for the Simple-LSTM model.
Table 9. Results for Simple-LSTM.

<table>
<thead>
<tr>
<th></th>
<th>Training Set RMSE (ft)</th>
<th>Testing Set RMSE (ft)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>18.653</td>
<td>16.658</td>
</tr>
<tr>
<td>$x$</td>
<td>1.725</td>
<td>1.614</td>
</tr>
</tbody>
</table>

In the training set, the average deviation for the longitudinal position of the trajectory was 18.653 ft, while the lateral position averaged a deviation of 1.725 ft. In contrast, the testing set displayed a longitudinal deviation of 16.658 ft and a lateral deviation of 1.614 ft for the trajectory.
Considering that the standard width of a highway lane is approximately 12 ft, the deviations observed in the testing set are relatively more acceptable. The discrepancies in the training set’s errors are notably greater than those in the testing set. Ideally, the error margins between the two datasets should be similar. Further validation and analytical exploration are warranted to enhance the model’s performance.

Regarding the graphical representation, the $y$ axis denotes the longitudinal position, while the $x$ axis represents the lateral position. The blue line traces the actual trajectory, and the red line depicts the predicted trajectory, which aligns well with the actual data.

Tables 10 and 11 present the results of the two distinct models.

### Table 10. Results for LC-LSTM.

<table>
<thead>
<tr>
<th>LC-LSTM RMSE</th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual (ft)</td>
<td>Normalized</td>
</tr>
<tr>
<td>$y$</td>
<td>7.811</td>
<td>$5.0337 \times 10^{-3}$</td>
</tr>
<tr>
<td>$x$</td>
<td>0.444</td>
<td>$4.9811 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

### Table 11. Results for LK-LSTM.

<table>
<thead>
<tr>
<th>LK-LSTM RMSE</th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual (ft)</td>
<td>Normalized</td>
</tr>
<tr>
<td>$y$</td>
<td>11.030</td>
<td>$7.1081 \times 10^{-3}$</td>
</tr>
<tr>
<td>$x$</td>
<td>0.752</td>
<td>$8.4409 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

When contrasting the two models, the LK-LSTM model demonstrated lower errors in both positions on the testing set compared with the LC-LSTM model. However, the LK-LSTM model exhibited higher errors in both positions on the training set than the LC-LSTM model. Overall, the performances of the two models are comparable. It is worth noting that lane changing typically occurs over a few seconds; hence, the data available for the LC-LSTM model may be limited. Gathering more lane-changing data could contribute to the development of a more accurate and robust model.

In relation to the Simple-LSTM model, the errors in the two specific models (LK-LSTM and LC-LSTM) were diminished. This enhancement in performance underscores the value of employing specialized models tailored to specific datasets.

### 3.3. Integrated Prediction Model

The optimal models identified from prior sections were integrated into a unified prediction model. For individual vehicles, the maneuver at each time step was initially predicted using the linear SVC model. If the prediction yielded a result of zero, then the LK-LSTM model was employed to forecast the longitudinal ($pos_y$) and lateral ($pos_x$) positions. Otherwise, the LC-LSTM model was utilized for position predictions. This approach enables the reconstruction of a vehicle’s complete trajectory.

For comparison, trajectories were also reconstructed using the Simple-LSTM model, serving as a baseline. The actual positional data, spanning 10,000 time steps, was employed to assess model performance. Table 12 presents a comparative analysis between the Simple-LSTM and Integrated-LSTM models.

### Table 12. Results for LK-LSTM.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Simple-LSTM</th>
<th>Integrated-LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$ RMSE (ft)</td>
<td>10.576</td>
<td>16.621</td>
</tr>
<tr>
<td>$x$ RMSE (ft)</td>
<td>1.870</td>
<td>1.401</td>
</tr>
<tr>
<td>Normalized RMSE</td>
<td>$6.815 \times 10^{-3}$</td>
<td>$10.711 \times 10^{-3}$</td>
</tr>
<tr>
<td>Normalized RMSE</td>
<td>$20.987 \times 10^{-3}$</td>
<td>$15.731 \times 10^{-3}$</td>
</tr>
</tbody>
</table>
In an evaluation of model accuracy, Figures 8 and 9 present the actual and predicted vehicle positions obtained using both the Simple-LSTM and Integrated-LSTM models, respectively. Figure 8 specifically focuses on the point-wise comparison of the true and estimated positions for several vehicle examples. Figure 8 displays the actual and predicted vehicular positions for IDs 31, 54, 90, and 100. The left column features the predictions generated by the Simple-LSTM model, whereas the right column showcases the results obtained via the Integrated-LSTM model. The first row of the figure is dedicated to illustrating the relationship between time and longitudinal positions, while the second row focuses on time versus lateral positions. Within each subplot, the red line represents the predicted positions, while the blue line delineates the actual positions.

Figure 8. Simple-LSTM and Integrated-LSTM predicted position for vehicles.

Figure 9 extends the analysis by depicting the full trajectories under both modeling paradigms. Figure 9 also displays the actual and predicted vehicular trajectories for IDs 31, 54, 90, and 100. The left figure features the predictions generated by the Simple-LSTM model, whereas the right figure showcases the results obtained via the Integrated-LSTM model, wherein the red line represents the predicted trajectory while the blue line delineates the actual trajectory. A closer examination of the figures reveals distinct differences in performance between the two models. Notably, the Integrated-LSTM model exhibited a greater variance in longitudinal positioning compared with its Simple-LSTM counterpart. On the other hand, the variance in lateral positioning was notably lower when using the Integrated-LSTM model. These observations are consistent with the quantitative findings reported in terms of the root mean square error (RMSE).
When comparing the performance of the Integrated-LSTM model with that of the Simple-LSTM model, a trade-off between the accuracies of the lateral and longitudinal positioning became evident. Specifically, the Integrated-LSTM model demonstrated reduced error in estimating $x$ but an elevated error in predicting $y$. This observation suggests a complex interplay between the two output variables that merit further investigation. Additionally, it is important to note that the model’s architecture involves a sequential two-step prediction process. Consequently, the accuracy of the second prediction step is inherently influenced by the precision of the initial step. This cascading effect of the error resembles a Bayesian decision-making scenario, where inaccuracies in early stages may be amplified in subsequent steps, compared with what might be observed in standalone models.

4. Conclusions

In the rapidly advancing domain of autonomous driving, the ability to predict lane changes accurately is of paramount importance. Such predictions not only enhance the capabilities of advanced driver-assistance systems (ADASs) but also play a pivotal role in ensuring road safety and optimizing traffic flow.

To address this challenge, we developed an integrated prediction model rooted in lane change maneuver classification. Utilizing the capabilities of long short-term memory (LSTM), a subset of recurrent neural networks, our approach prioritized the maneuver classification model. This model was designed to predict the vehicle’s driving behavior (either lane changing or lane keeping). Depending on the predicted maneuver for each
timestep, a corresponding trajectory prediction model was applied, ensuring a comprehensive prediction of the vehicle trajectory.

Our integrated prediction model demonstrated commendable accuracy with the majority of our trained models. This robust performance in maneuver classification significantly enhanced the reliability of the subsequent trajectory prediction model, which was tailored for specific driving behaviors like lane changing and lane keeping.

While our study represents a meaningful advancement in the field, it also highlights several avenues that merit additional investigation. One area of concern is our existing maneuver classification approach, which leverages the lateral speed and steering angle as key indicators for labeling maneuvers. This strategy occasionally generates labels that may not accurately capture the nuances of actual driving behaviors. Although our model is theoretically well suited for real-world applications, we currently lack a real-time testing environment. We aim to remedy this by deploying our methodology in authentic road conditions in future research. Another limitation pertains to the data set utilized. Our experiments relied on the NGSIM dataset, which does not encompass a broad spectrum of extreme or varied road conditions, such as an emergency road vehicle approaching from behind, an accident happening in the same lane, and the occurrence of vehicle failure that pushes one to switch lanes. For future endeavors, we plan to incorporate more diverse and comprehensive datasets to substantiate our algorithmic validations.

Author Contributions: Conceptualization, H.L. and K.W.; methodology, H.L.; software, K.W.; validation, K.W., S.F. and H.S.; formal analysis, H.X.; investigation, H.S.; data curation, K.W.; writing—original draft preparation, H.L.; writing—review and editing, K.W.; visualization, S.F.; funding acquisition, H.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.


Conflicts of Interest: The authors declare no conflict of interest.

References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.