Lateral Classification of Vehicle Behavior for Automated Logic Scenario Generation

Vjekoslav Diklić[a,*], Ivan Marković[b]

[a] *dSPACE d.o.o.,*
Ul. grada Vukovara 284, HR-10000, Zagreb, Croatia,
[b] *University of Zagreb Faculty of Electrical Engineering and Computing,*
Department of Control and Computer Engineering,
Laboratory for Autonomous Systems and Mobile Robotics,
Unska 3, HR-10000, Zagreb, Croatia,

Abstract

Advanced testing and validation in the automotive industry are predominantly conducted through real-world driving, while simulation is primarily used for scenarios that are difficult, dangerous, or impractical to reproduce on the road. The production of a well-designed simulation scenario includes numerous edge cases and relies on expert labor, but it can still miss real-world complexities. Given that, scenarios based on real-world driving data are more desirable, but are dominantly based on replaying the positions of all vehicles over time. For greater flexibility, real-world driving scenarios should be automatically converted into logic scenarios with adjustable segments such as lane keeping, lane changing, accelerating, etc. This paper introduces a new method for detecting and classifying lateral vehicle movements. The method is based on polynomial fitting and identifies the start and end points of vehicle lane changes with inflection points of a third-degree polynomial. For evaluation, two publicly available, manually labeled lane-change datasets are used, and a novel highway lane-change simulation dataset is introduced. The polynomial approach is compared against a bidirectional long short-term memory (Bi-LSTM) model introduced in this work, as well as three additional neural network architectures drawn from recent literature, followed by a discussion of the experimental validation for each.

Keywords: Logic Scenarios, Driving Scenario, Segment Identification, Automated Scenario Generation, bidirectional long-short term memory network, Lane Change Classification

1. Introduction

In the modern automotive industry, the complexity of road vehicles is continually rising, particularly with the integration of autonomous driving and other Advanced Driver Assistance Systems (ADAS) functions. Systems like ADAS [1] and traffic self-awareness [2] are data-driven and require a significant amount of data, which can be supplemented or replaced by data acquired in simulations. Additionally, there is a special focus on advanced testing and validation, where simulations play a crucial role and are recognized as standard practices in the automotive industry [3]. These simulations are implemented throughout all stages of vehicle development. Testing and validation conducted in simulations offer numerous advantages, including repeatability, expedited execution time, cost efficiency, and other valuable attributes. Given that modern vehicles are intricate and complex machines intended to operate in diverse weather, road, and traffic conditions, it becomes essential to test these conditions in simulations. To encompass the full range of possible combinations of the aforementioned conditions, the term *scenario* is used, as proposed by Ulbrich et al. [4], along with its sub-elements *scene* and *situation*, whereas mathematical definitions have been proposed by Andreotti et al. [5]. Simulation scenarios can be generated either manually, based on prior knowledge and experience, or by converting recorded real-world driving data, as described in the works of Menzel et al. [6, 7]. The scenarios generated from real-world data hold greater value for testing and validation purposes. However, due to their specific lower abstraction level, they enable the replay of only the exact same driving situation as in the original data. Hence, they are referred to as *replay* or *concrete* scenarios [6]. On the contrary, simulation scenarios manually created using high-level abstraction description languages, such as OpenSCENARIO [8], or logic segments [9], known as *logic* scenarios [6], offer a higher level of abstraction. This abstraction enables flexibility in modifying almost any parameter within the simulation scenario, thereby expanding the scope of testing. Manually adding higher abstraction levels is feasible but inefficient, demanding a substantial amount of effort. This holds particularly true, given that the majority of tests necessitate a high number of simulation scenarios. Therefore, in order to address a high number of scenarios, an automatic conversion method is the only approach to tackle such a problem. The primary challenge in the automatic conversion process lies in the initial step, which involves the classification of vehicle actions based on the given data [10].

Zhao et al. [11] proposed the conversion of real-world driving data through automated classification algorithms based on big data and ontology [12]. Authors in [7] defined a *data-driven*

*Corresponding author: Vjekoslav Diklić
Email addresses: vjekoslav.diklic@gmail.com (Vjekoslav Diklić), ivan.markovic@fer.hr (Ivan Marković)

approach for generating simulation scenarios for testing and validation with ISO 26262 [13] as the reference. A more practical approach was introduced by Lages et al. [14] and Made et al. [15], who based their automated conversion on data from laser sensors. Furthermore, numerous approaches are grounded in machine learning algorithms [10, 16, 17, 18, 19, 20, 21]. Certain authors, such as Hülnhagen et al. [22], employed fuzzy logic, while a hybrid method integrating neural networks and fuzzy logic was introduced by Lin et al. [23]. Sonka et al. [24] employed probabilistic and fuzzy classification, thereby enhancing the accuracy of the methods. This paper presents a novel method for detecting and classifying the lateral movement of vehicles and compares its performance with an earlier approach based on neural networks, as published in the preliminary work [25]. The motivation for developing a novel method based on polynomials and evaluating its performance against a neural network model stems from the challenges of training the neural networks for this particular problem. Real-life datasets that match the requirements are frequently scarce, thereby restricting the practicality of their applications. The novel method introduced in this study is based on polynomial fitting and provides adaptability and parametrization, eliminating the need for dedicated training data. Furthermore, new lane change classification metrics are proposed, motivated by the need for a more detailed evaluation of model performance. Standard metrics are not sufficient to describe quality of lane change parameters. Metrics like the accuracy and confusion matrix can describe how well an estimate matches a label, but in scenario generation, parameters such as symmetry of the estimate, continuity, and position have more practical value. Finally, a new publicly available dataset, HLC-Sim [26], is introduced. It consists of vehicle trajectories and ground truth lane change annotations for highway driving scenarios, and is currently the only dataset containing such labels. The dataset contains 2912 unique driving trajectories and was generated using the dSPACE ASM simulation toolset [27]. The polynomial-based method and the introduced *Bi-LSTM* model were evaluated on the new *HLCsim* [26] dataset, as well as the established *NGSIM* [28] and *HighD* [29] benchmarks. For comparison, the *Dual Attention Bi-LSTM* model by Ashfaq et al. [19], the maneuver-classification stage of the LK-/LC-LSTM framework by Liu et al. [20], and the sequence-to-sequence *TTS* model by Wan et al. [21] were also implemented. All models had only their input and output layers adjusted to match the data dimensions, ensuring a fair and consistent evaluation across all approaches. Although the neural network models have resulted in slightly better performance, the proposed polynomial method does not require training and can operate with various datasets, requiring only minor parameter adjustments. Moreover, due to the nature of how the proposed method functions, it produces more continuous output, minimizing abrupt fluctuations compared to the neural network model.

The paper is structured as follows. Section 2 provides a comprehensive description of what constitutes a driving scenario and addresses the challenge of increasing the abstraction levels in scenario descriptions. The proposed polynomial-based method is introduced in Section 3, while the neural network models and their training are covered in Section 4. Dataset forming and preprocessing are elaborated in Section 5, while the neural network training process, the proposed metrics, and the evaluation results are thoroughly discussed in Section 6. Finally, Section 7 concludes the paper.

2. Problem Statement

The most straightforward approach to describing object movement in a simulation is through trajectory-based movement, which involves representing the position of the object over time. This approach is frequently utilized when converting real-world driving into a simulation scenario. The abstraction level for such a *replay* or *concrete* [6] simulation scenario description is very low, as it cannot facilitate easy modification of scenario parameters, such as the velocity or lane position of the vehicle. These parameters are crucial for testing multiple vehicle systems, such as Automatic Emergency Braking (AEB) [30], Adaptive Cruise Control (ACC) [31], and Lane Keeping Assistance (LKA) [32]. In contrast to replay scenarios, *logic* scenarios describe vehicle behaviour through the use of conditional interconnected parameterized segments that are human-readable. Such an approach simplifies the aforementioned modifications to vehicle behaviour, providing a more flexible and comprehensible representation. These are then used during simulations as the input to a simulated physical model of the vehicle, resulting in vehicle behaviour that is the result of evaluating the vehicle model with input values, rather than following a trajectory without any physical properties.

The parametrization of segments can exhibit significant variation depending on the simulation tool and testing environment in use. Commercial simulators like dSPACE ASM ModelDesk [27] and IPG CarMaker [33] describe segments and their transition conditions in flow charts. CarSim [34] and rFpro [35] simulators offer scripting solutions for describing driving scenarios. Open-source simulators like CARLA [36] also support an open language for describing scenarios, such as OpenSCENARIO [37], developed by ASAM [38].

With a segment-based approach, describing more complex vehicle driving maneuvers is achieved by conditionally binding multiple segments. An example of driving with a constant speed of 50 km/h for 60 seconds and then braking to a full stop can be described using two segments. The first segment represents driving at constant speed, and the second segment represents the braking phase. These two segments are connected by a time-conditioned execution of the first segment for 60 seconds, while the second segment's execution is conditioned on the vehicle velocity reaching 0 km/h.

In order to support as many simulation tools as possible, this paper addresses only the basic logic scenario segments, grouped into two categories: lateral and longitudinal segments. Vehicle longitudinal actions are described with three types of segments: *Braking*, *Accelerating*, and *Constant Velocity*. These segments are based on constant acceleration, and by combining multiple segments with different values of acceleration, it is possible to describe most longitudinal vehicle movements, as illustrated in Fig. 1. Lateral actions of vehicles are described using two types of segments: *Lane keep (LK)* and *Lane change (LC)*. Further,

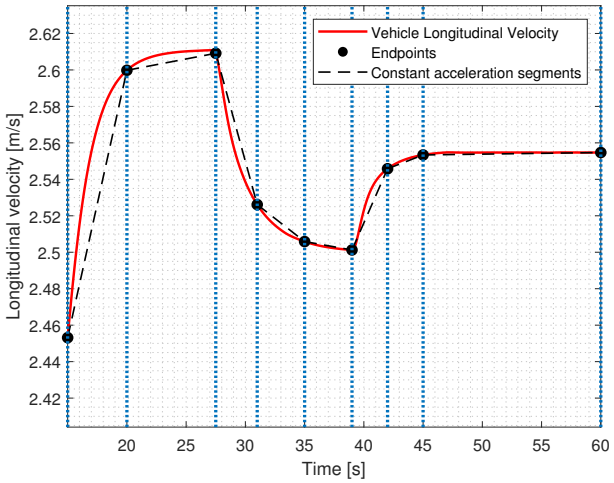


Figure 1: Longitudinal velocity segments

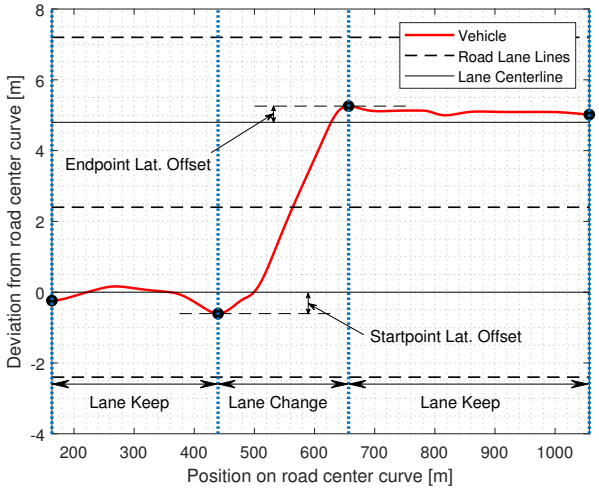


Figure 2: Lane change example

lane change can be subdivided by direction into two subsegments: *Lane change right (LCR)* and *Lane change left (LCL)*, as depicted in Fig. 2. Parallel to longitudinal segments, lateral segments also encompass parameters such as lateral acceleration, start lane, end lane, lateral deviation from the road lane centerline, and other pertinent factors. The combination of lateral and longitudinal segments can describe most highway situations.

Considering this, the conversion of replay scenarios to logic scenarios provides a way to obtain a large number of logic scenarios from real-world situations. However, this process demands the precise classification of parts of the replay scenario into logic segments. It necessitates considering both the vehicle trajectory and the road to appropriately convert the complete scenario.

The objective of the conversion process is to describe vehicle movement through the utilization of descriptive segments interconnected with conditions. Following the simulation, the resulting vehicle trajectory movement, based on these segments, needs to closely match the original trajectory as accurately as possible. It is crucial to note that the outcome of the conversion process is not deterministic, as a single replay scenario can be described in multiple ways within the logic scenario context. The conversion process can be divided into two steps: classification and parametrization. An overview of this process for vehicle lateral movement is shown in Fig. 3. The classification process is required to mark all actions along with their beginning and end, encompassing both longitudinal and lateral actions. The parametrization process heavily depends on the quality of the classification output. Each classified action corresponds to a single logic description segment, accompanied by its associated parameters. Moreover, parametrizing lateral and longitudinal segments that overlap either in time or position can be challenging, and additional segments may be necessary to address this issue. Longitudinal classification is beyond the scope of this article. Therefore, only lateral classification and parametrization will be discussed, and longitudinal movement will be considered constant.

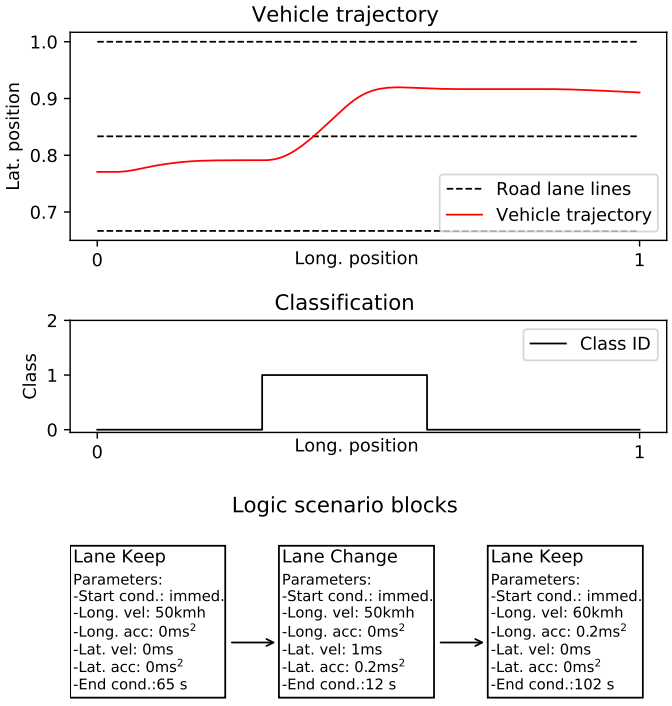


Figure 3: Conversion process overview

3. Proposed Polynomial Lane Change Classification Method

The polynomial method is an analytical approach to classifying lateral vehicle movement in trajectories represented in the *S–D* coordinate system. Such an approach offers an alternative to the most common solution, namely neural networks. The method proposed in this section is inspired by the works of Nelson et al. [39], who modeled vehicle movement with a polynomial, and Zhou et al. [40], where vehicle lane changes were modeled using a hyperbolic tangent function and compared to a quintic polynomial model. In the polynomial method, lane change classification involves fitting a third-degree polynomial

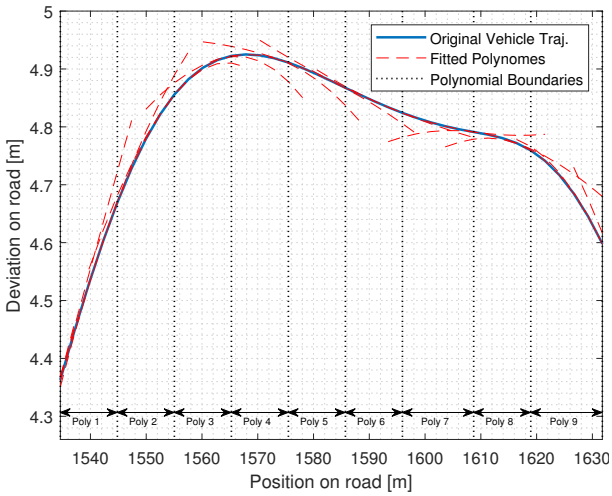


Figure 4: Vehicle trajectory conversion to a polynome

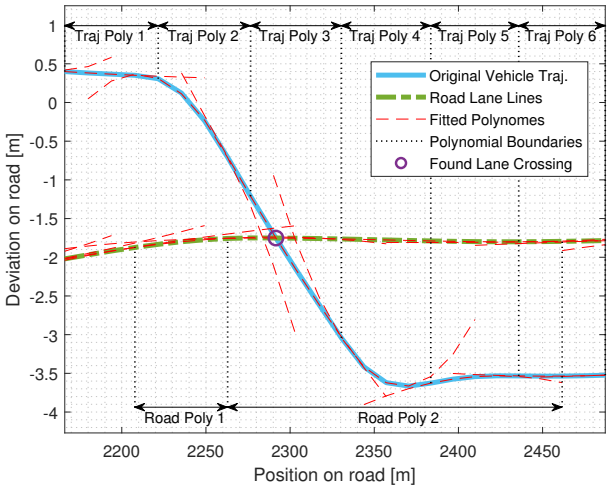


Figure 5: Trajectory lane line crossing position detection.

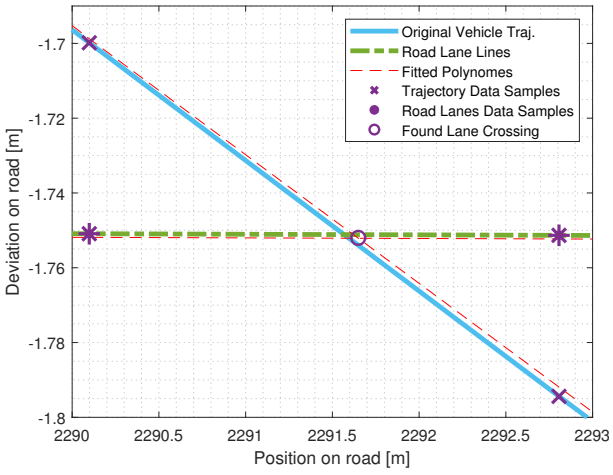


Figure 6: Close up on the trajectory lane line crossing position detection.

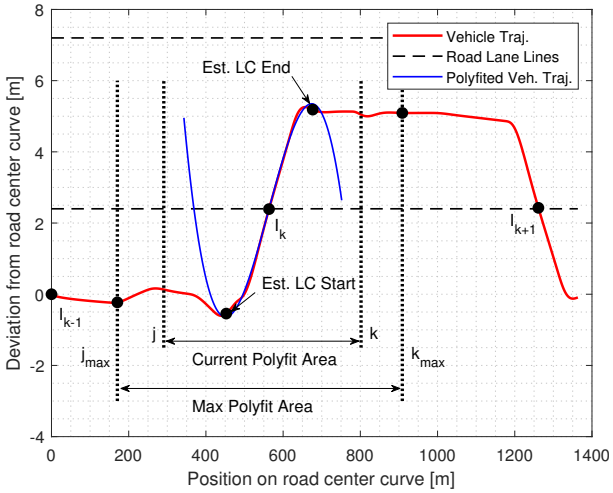


Figure 7: Estimation of lane change starting and end points.

around the position where the vehicle crosses the lane line. The polynomial method can be divided into two phases: (i) finding the lane line crossings and (ii) determining the lane change start and end points. To unambiguously find lane line crossing points, despite noise, uneven sample rate, and other unwanted artifacts in the data, lane line crossings are calculated as the intersection of two polynomials. It is well known that complete vehicle trajectories and road lane lines cannot be accurately represented using a single polynomial of any degree. Thus, the solution is to create a family of polynomials that approximate the trajectory or road lane lines, where each family member represents only a segment of the input data. The criterion function for determining the size of each segment, for a specific polynomial, is calculated using the following formula:

$$J = \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_{poly} - Y_{input})^2}{\log_e(\frac{n}{2})}. \quad (1)$$

An example of a vehicle trajectory estimated with a family of polynomials can be seen in Fig. 4. Lane line crossings are detected by comparing each lane line polynomial family to the vehicle trajectory family and checking whether the two polynomials intersect. This approach allows the detection of lane changes with subsample precision, since both road lane lines and the vehicle trajectory are converted to a continuous representation. Figure 5 illustrates a simplified example of the detection process. The dashed lines in the figure represent the fitted polynomials for both the vehicle trajectory and the road lane lines. The range of each polynomial fitting is indicated by black arrows. In this example, a lane line crossing is detected at the intersection of polynomial parts for the road marked as "Road Poly 2" and the vehicle trajectory marked with "Traj Poly 3". Even though this process can be computationally demanding, this is not a limitation, as the algorithm's application is strictly offline. Processing a 15-minute driving scenario can take up to 60 minutes, depending on the number of road lanes and vehicles involved. As mentioned earlier, the advantage of this approach is its ability to work with subsample precision, enabling the detection of lane line crossings even when there are no overlapping samples, as shown in Fig. 6. The output in this phase is a list of indices that correspond to lane line crossings in the data. The subsequent step involves determining the start and end points

of the lane change. This is achieved by fitting a third-degree polynomial lane change model around the detected lane line crossing, as illustrated in Fig. 7. However, this fitting process is not straightforward, as the fitting range starts around the crossing point in the center and iteratively expands outwards. The range expansion is limited by parameters k_{max} and j_{max} which represent the midpoints between the current lane line crossing point I_k and an earlier I_{k-1} or the next I_{k+1} one, respectively. Parameters k_{max} and j_{max} are calculated using the following formulas:

$$j_{max} = \frac{I_k - I_{k-1}}{2}, \tag{2}$$

$$k_{max} = \frac{I_{k+1} - I_k}{2}. \tag{3}$$

The range is iteratively extended by parameters j and k, which are limited by j_{max} and k_{max},and continue expanding until both reach their limits. If one limit is reached earlier than the other, the remaining parameter continues increasing in the next iterations. After the current area is fitted by a third-degree polynomial function (4), the obtained parameters are used to calculate the inflection points ($LCIs$) (5):

$$f(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0, \tag{4}$$

$$LCI_{1,2} = \frac{-a_2 \pm \sqrt{a_2^2 - 3a_3a_1}}{3a_3}, \tag{5}$$

where $LCI_{1,2}$ must satisfy:

$$\forall LCI_{1,2} \in: \quad \mathbb{R} \bigcap \langle j_{max}, k_{max} \rangle. \tag{6}$$

The subsequent step involves proceeding and computing the value of the criteria function.

$$J_i = \frac{\frac{1}{k+j}\sum_{i=1}^{k+j}(Y_{3rdpoly} - Y_{Poly})^2}{\log_e(k+j)}, \tag{7}$$

This value is then compared to the previous one (initially set to $+\infty$) and the algorithm continues until convergence.

The type of a lane change can be determined by comparing the values of the vehicle trajectory at $LCI_{1,2}$.

$$LCT = Y(\min(LCI_{1,2})) - Y(\max(LCI_{1,2})) \tag{8}$$

$$\text{sign}(LCT) = \begin{cases} 1: & \text{Lane Change Right} \\ -1: & \text{Lane Change Left} \end{cases} \tag{9}$$

Lane change start points are always the smaller root solutions $\min(LCI_{1,2})$ as can be seen by examining (5), while the end points are the larger root solutions $\max(LCI_{1,2})$. Apart from performing classification, the estimated polynomial parameters a_3, a_2, a_1 and a_0 can also be used to compute the velocity and acceleration of the lane change.

4. The Neural Network Method

Neural networks have emerged as a prevalent approach for addressing classification problems [41]. Motivated by preliminary work [25], which was extended in this paper, a neural network approach is proposed as a comparative method, contrasted with the polynomial method. The optimal model, derived from preliminary work involving multiple network configurations for this specific task, was chosen as the starting point. To enhance its performance, an expansion was implemented by introducing a direct connection from the output of each convolutional layer to the input of the *Bi-LSTM* layer, as shown in Fig. 8. An LSTM (Long Short-Term Memory) layer utilizes gated units with self-connected cells, allowing it to capture and propagate information over extended sequences while mitigating the vanishing gradient problem. Furthermore, the Bi-LSTM layer employs two LSTM networks, one processing input data in forward order and the other in reverse order, enabling it to capture context from both past and future inputs. In addition, an extra Bi-LSTM layer was added, followed by two fully connected layers. The input of the model is $n \times 10$, where n represents the number of data samples, while the second dimension consists of the vehicle's S,D coordinates followed by 8 lane lines. The initial three convolutional layers are responsible for extracting features from the input data. Each convolutional layer comprises 32 filters, with the first layer employing a kernel size of 5, and the subsequent two utilizing a kernel size of 2. By merging the input and output of all convolutional layers, the data size increases to $n \times 10$. This amalgamation provides the subsequent layers with a more comprehensive view of both the input data and the features extracted by the convolutional layers. The two Bi-LSTM layers consist of 128 cells in the backward layer and 128 cells in the forward layer, resulting in an output of $n \times 256$. This layer gathers all the available data and prepares it for further processing in the subsequent layers of the model, which consist of two fully connected layers. The first of the two fully connected layers has 64 neurons, and the second layer has 32 neurons. The model's output is formed by a 3-neuron softmax layer, yielding an output

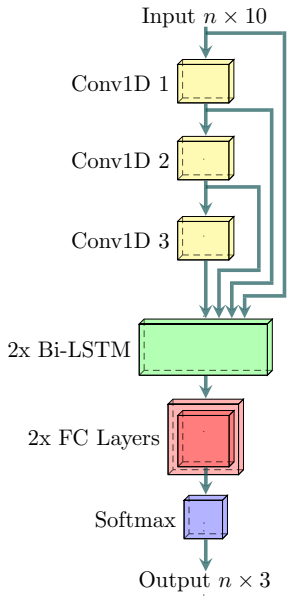


Figure 8: The proposed neural network model with multiple consecutive convolution layers and Bi-LSTM

size of $n \times 3$. In this context, n signifies the number of samples, and the second dimension gives the probability for each class.

The *Dual Attention* model [19], the maneuver-classification stage of the LK-/LC-LSTM [20], and the sequence-to-sequence *TTS* model [21] were also implemented for comparison. All models had their input and output layers adjusted to match the dataset dimensions, ensuring a fair and consistent evaluation across all approaches. *Dual Attention* [19], except for input and output size adjustment, was not changed in any other respect. *TTS* [21] was modified so that the internal encoder-decoder LSTM produces a 3-class softmax output for variable-length inputs, replacing the original fixed-window binary classifier. Due to dataset limitations, only the maneuver-classification stage of the LK-/LC-LSTM framework [20] was used. In this form, it is a simple neural network with two feed-forward hidden layers, along with input and output layers, and is hereafter referred to as *Maneuver-Classification*.

All aforementioned models were trained using the Adam optimizer [42] and the categorical cross-entropy loss function. The training and validation of the models were performed using the dataset shown in Table 2. Models were trained for 500 epochs, and those that achieved the best validation accuracy were selected for further analysis. Metrics for each model during training are shown in Figure 9, while the final training results are presented in Table 1. The metrics from the training process show that the *Bi-LSTM* model exhibits strong and consistent performance with minimal overfitting. It achieved the highest final training accuracy (99.847%), while its best validation accuracy ranked second (97.416%). The low standard deviation in validation accuracy (0.01080) indicates stable training and reliable generalization. The Dual Attention model achieved the highest validation accuracy (97.586%) with a moderately high standard deviation (0.02660). Despite fluctuations, the model converged well, with a final validation accuracy of 97.334% and a training accuracy of 98.462%, indicating mild overfitting and overall good generalization. The *TTS* model achieved high and stable performance, with the best validation accuracy of 96.722%. Training accuracy closely followed, ranking second with 96.804%, and the final accuracies were 97.739% (train) and 96.340% (validation). The validation accuracy standard deviation (0.01869) reflects a well-behaved training process with slight overfitting toward the end. The *Maneuver-Classification* model exhibits the lowest performance among the group but maintains consistent behavior. Training and validation accuracies are nearly identical (84.836% and 85.277%, respectively), indicating no overfitting. The results confirm limited model capacity due to the shallow and simple model architecture.

Table 1: Validation and training performance metrics for each model at peak and final epochs

Model	Epoch	Val Acc	Val Loss	Train Acc	Train Loss
Bi-LSTM	112	0.97573	0.08536	0.98675	0.03165
	500	0.97416	0.24860	**0.99847**	**0.00368**
Dual_Att	361	**0.97586**	**0.07962**	0.98116	0.04703
	500	0.97334	0.09793	0.98462	0.03848
TTS	176	0.96722	0.10217	0.96804	0.08665
	500	0.96340	0.14479	0.97739	0.06131
Man_Class	480	0.85537	0.38463	0.84820	0.40418
	500	0.85277	0.38591	0.84836	0.40341

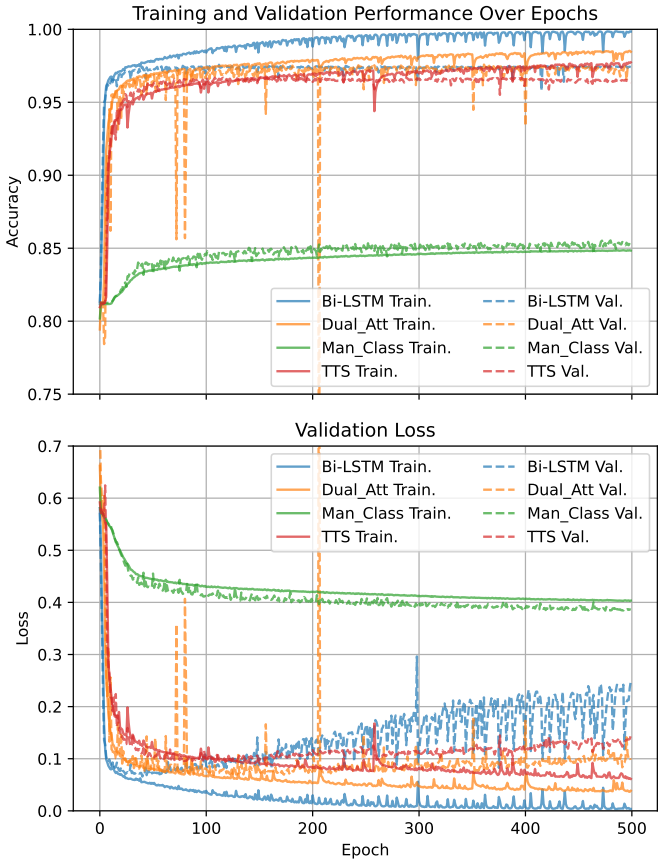


Figure 9: Validation accuracy and loss during learning process

5. Dataset Compilation

Due to the limited availability of datasets containing annotated lane change actions, a novel dataset was curated and generated specifically for learning and evaluation. The dataset is the result of combining three datasets: HighD [29], NGSIM [28], and HLCsim [26]. Since the HighD and NGSIM datasets do not contain labels for lane change actions, labels had to be added manually. The manual labeling of lane change actions involved placing markers in approximate areas corresponding to the start and end positions of the lane changes. The encoding of label classes was defined as follows:

- Lane keep: 0
- Lane change left: 1
- Lane change right: 2

While this method of data labeling may introduce minor imprecisions, it is acceptable given the non-deterministic nature of

Table 2: Dataset composition and augmentation

Description	**Dataset:**	**HighD**		**NGSIM**		**HLCsim**	
	type:	learn	val.	learn	val.	learn	val.
Initial	LK	2266	551	496	121	634	158
	LCL	2353	578	477	118	776	199
	LCR	1939	483	194	55	921	224
After mirroring	LK	4532	551	992	121	1268	158
	LCL	4292	587	671	119	1697	199
	LCR	4292	483	671	55	1697	224
After adding noise	LK	18128	571	3968	121	5072	158
	LCL	17168	578	2684	119	6788	199
	LCR	17168	483	2684	55	6788	224
Sum		52464	1632	9336	295	18648	581
Learn/valid [%]		65.21	65.07	11.61	11.76	23.18	23.17
Total num. of traj.		80448	2508				

the conversion process. The HLCsim dataset already contains ground truth lane change labels generated by the simulation, which utilized lateral segments. The combination of the three datasets introduced a minor compromise in terms of label precision; however, it brings the advantage of using a blend of realistic and synthetic data for training and validation purposes. To broaden the dataset, the dataset was enhanced to improve data diversity and the noise robustness of the trained model through augmentation [43]. Augmentation was applied in two steps, with the first step involving the mirroring of trajectories from left to right. This mirroring process produces the opposite lane change of the original, ensuring an equal number of lane change left and lane change right classes. In the second augmentation step, noise was applied at different signal-to-noise ratios: 70 dB, 50 dB, and 20 dB. The augmentation steps were exclusively applied to the training portion of the dataset, initially divided into an 80:20 training/validation ratio. The dataset's final size comprises 80448 trajectories for training and 2508 for validation; details on data composition and augmentations are shown in Table 2. To simplify data handling, the data from the aforementioned datasets was converted to the curvilinear coordinate system, also known as the S–D system. This system, as described in [44], represents the position on the curve as S (arc length along the curve) and the lateral deviation from the curve center as D. In order to perform the conversion to the S–D coordinate system, a road centerline was selected as the reference curve. This conversion to the S–D system provides a data representation that does not include information about road curvature, which simplifies further calculations. After conversion to the S–D system, data was scaled to range $[0, 1]$. The scaling of the data was done to limit the range and avoid the need for batch normalization steps in the neural network model. This approach ensures that both methods have the same data representation, allowing for fair and consistent comparisons. An example of data converted to the S–D coordinate system and scaled to range $[0, 1]$ can be seen in Fig. 10. The narrowing and winding of lanes observed in this figure represent the closing

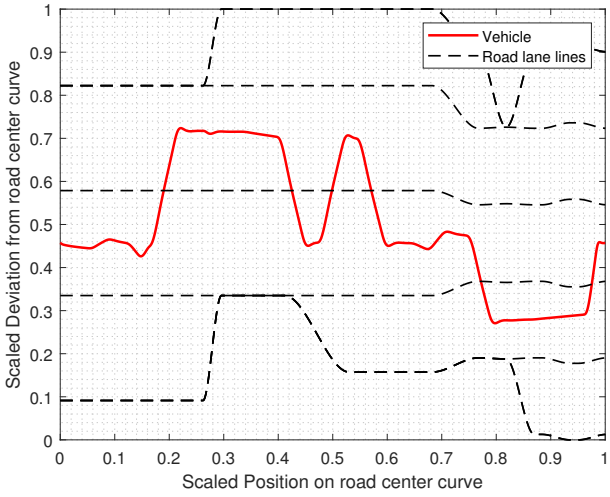


Figure 10: Complete scenario represented in the scaled S–D coordinate system.

and opening of road lane lines, such as highway entrances and exits.

6. Results and analysis

6.1. Classification assessment

In classification tasks, the standard approach for defining metrics involves utilizing the confusion matrix and accuracy expressions. For vehicle behavior classification, authors such as Sonka et al. [24] used the confusion matrix, while Schlechtriemen et al. [45] and Zekany et al. [46] used the true positive rate and false positive rate. The works in [10, 47, 22] used only accuracy. Zhang et al. [17] used both the confusion matrix and accuracy to interpret their results. However, due to the specific way in which the output is used, accuracy and the confusion matrix alone are not sufficient to fully describe classification quality. To improve the quality assessment of the classification output, three additional metrics are proposed: *Center Offset*, *Start/End*, and *Estimate Usability*.

The first metric, referred to as the *Center Offset*, represents the difference in the position of the centers of the lane change between the label and the estimate. When a logic simulation scenario is generated using the classification output with a low value of *Center Offset*, the positional precision during the lane change may be reduced. The *Center Offset* is calculated as a percentage by computing the distance error between the center of the labeled lane change C_{AB} and the center of the estimated lane change $C_{ab}$and dividing it by the number of samples in the scenario, N.

$$\text{Center offset} = \frac{|C_{AB} - C_{ab}|}{N}. \tag{10}$$

The *Start/End* metric represents the extent to which the label and estimate differ in their start and end positions. It quantifies the discrepancy between the start and end points of the label and the estimate and can be calculated with the following expression:

$$\text{Start/End} = 1 - \frac{|B - A + b - a|}{N}. \tag{11}$$

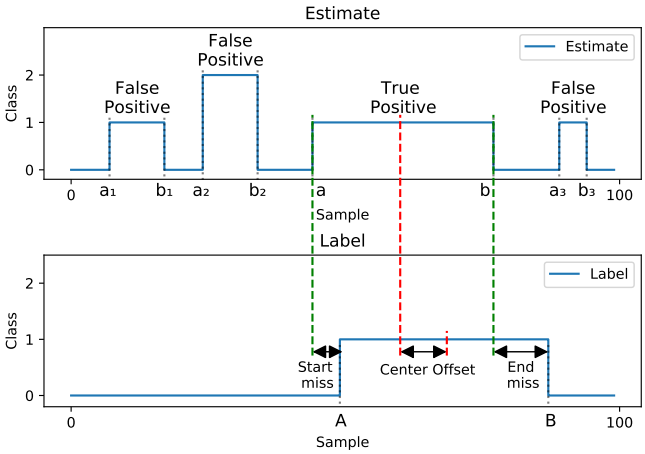


Figure 11: Parameters for classification assessment

A higher Start/End value indicates a better match between the label and estimate in terms of their respective starting and ending positions, while a lower value indicates greater deviation. The metric has a maximum value of 1, which represents a perfect match between the starting and ending positions of the estimate with the label. A logic simulation scenario based on classification output with a low Start/End value may start the lane change too early or too late.

The third metric, *Estimate Usability* quantifies the extent to which the estimate is "usable" for the reconstruction of logic scenario segments. It is defined as the ratio of false positives to the sum of all positives, as given by the following expression:

$$\text{Estimate usability} = \frac{b-a}{\sum_{i=1}^{n}(b_i - a_i) + (b-a)}. \tag{12}$$

A lower value of *Estimate Usability* indicates that some of the generated lane change segments in the simulation scenario may not be present in the input data. To better understand the metrics and their respective formulas, refer to Fig. 11. The lower part of Fig. 11 displays a labeled left lane change action, while the upper part illustrates an estimated detection example. Fig. 11 shows the parameters necessary for calculating the proposed metrics, including *Center Offset*, *Start/End* errors, and false positive detections of both the same and opposite types of lane changes. A common example of labels and estimates is shown in Fig. 12, while Table 3 displays the values for each of the metrics for the given examples, along with accuracy and confusion matrix parameters. The first two examples in Fig. 12 demonstrate how the *Start/End* metric describes early and delayed detection cases. It is noteworthy to observe in Table 3 that all metric values for these two examples are identical. This symmetry in the results is a consequence of the identically sized underestimate in Example 1 and overestimate in Example 2. The third example demonstrates how the *Center Offset* parameter behaves when the centers of the label and estimate are not identical. The fourth example illustrates how *Estimate Usability* reflects the false positives in the estimate, while changes in the *Center Offset* can be observed throughout all the given examples. In the fifth

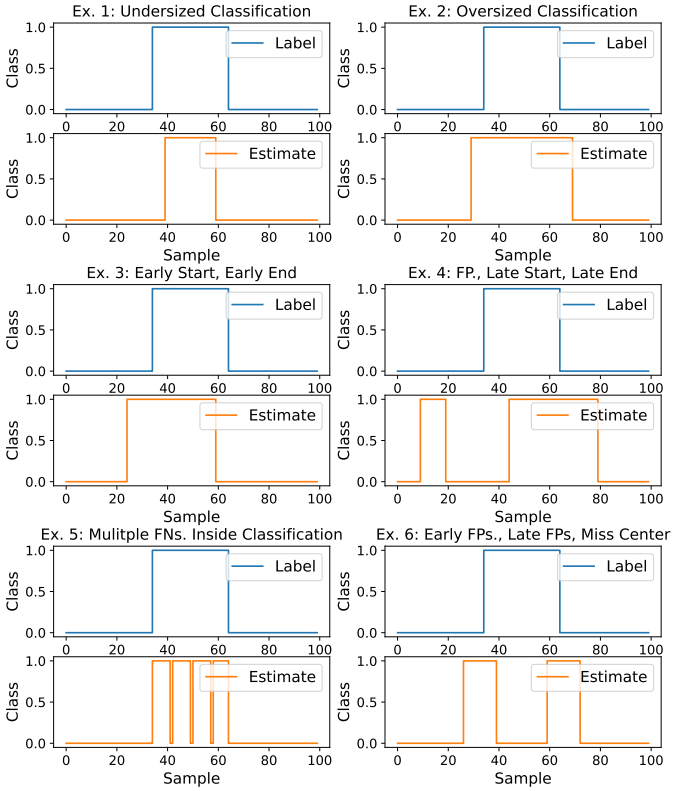


Figure 12: Classification examples

example, there are multiple detections where only one detection is expected. Depending on the specifics of the error, this can lead to the generation of multiple lane changes in a logic simulation scenario, which may result in the vehicle ending up in the wrong lane. The sixth example demonstrates how false detections are reflected in the metric values. Such errors in classification can result in multiple segments that describe the same action In practice, such errors can lead to various issues, ranging from the complete failure of the conversion process to uneven or unpredictable vehicle movement.

Table 3: Classification assessment on given examples

no.	Acc.	TP	TN	FP	FN	Start/ End	Center offset	Estimate usability
1.	0.9	0.3	0.633	0.033	0.033	0.9	1	1
2.	0.9	0.3	0.633	0.033	0.033	0.9	1	1
3.	0.85	0.283	0.617	0.05	0.05	0.95	0.925	1
4.	0.65	0.217	0.55	0.117	0.117	0.95	0.875	0.778
5.	0.97	0.323	0.657	0.01	0.01	0.77	0.965	0.259
6.	0.64	0.21	0.547	0.12	0.12 3	0.83	0.835	0.5

6.2. *Evaluation results*

All methods, including the polynomial-based method, the *Bi-LSTM* neural network, and models from other authors, were evaluated on the validation portion of the dataset, as outlined in

Table 4: Classification assessment on combined dataset

Noise level	Method	Accuracy	Start/ End	Center offset	Estimate Usability
	Poly	0.88250	0.89881	0.90619	0.87449
	Bi-LSTM	**0.97402**	**0.96759**	**0.97729**	**0.97965**
w/o	Dual_Att	0.97323	0.96572	0.97623	0.97881
	TTS	0.95691	0.88843	0.90112	0.88919
	Man_Class	0.85826	0.83517	0.88754	0.86644
	Poly	0.88476	0.89741	0.91668	0.89104
	Bi-LSTM	0.97395	0.96759	0.97729	0.97964
70 dB	Dual_Att	**0.97321**	**0.96572**	**0.97622**	**0.97880**
	TTS	0.95639	0.88687	0.90067	0.88291
	Man_Class.	0.85819	0.83388	0.88707	0.86347
	Poly	0.81050	0.85581	0.92432	0.77310
	Bi-LSTM	**0.97334**	**0.96826**	**0.97816**	**0.98073**
50 dB	Dual_Att	0.97245	0.96518	0.97591	0.97853
	TTS	0.95081	0.82029	0.84306	0.78808
	Man_Class	0.85666	0.79914	0.88143	0.73651
	Poly	0.15507	0.54076	0.64969	0.07850
	Bi-LSTM	**0.94385**	**0.80274**	**0.83076**	**0.74348**
20 dB	Dual_Att	0.93992	0.81405	0.83710	0.78942
	TTS	0.92416	0.81422	0.85347	0.78826
	Man_Class	0.83982	0.55767	0.65792	0.26185
	Poly	0.68321	0.79820	0.84922	0.65428
	Bi-LSTM	**0.96531**	**0.92937**	**0.94246**	**0.93236**
avg.	Dual_Att	0.96076	0.92771	0.94546	0.93110
	TTS	0.92598	0.78832	0.82569	0.70551
	Man_Class	0.83953	0.73820	0.82469	0.63005

Table 2. During the evaluation process, the data were corrupted with multiple levels of noise (70dB, 50dB, 20dB) to evaluate how each method would respond to such challenges. The evaluation results are shown in Table 4. The neural networks outperformed the polynomial-based approach, with the Dual Attention and *Bi-LSTM* models achieving the highest results. Without noise, the *Bi-LSTM* model achieved the best performance across all metrics. The polynomial-based method achieved better results than the *Maneuver-Classification* model across all metrics and outperformed the *TTS* model in all metrics except accuracy. By introducing noise corruption at a level of 70 dB, the Dual Attention model outperformed all others across all classification metrics, by a small margin closely followed by the *Bi-LSTM* model. The added noise did not change the ranking of the polynomial method, which still outperformed the *Maneuver-Classification* and *TTS* models in most metrics. Nevertheless, the polynomial method achieved better results under added noise across all metrics except Start/End. At the level of 50 dB, the *Bi-LSTM* model took the lead across all metrics, while the Dual Attention model ranked second. The polynomial method started to degrade under added noise, as all metrics decreased except for Center Offset. It was no longer able to match the performance of

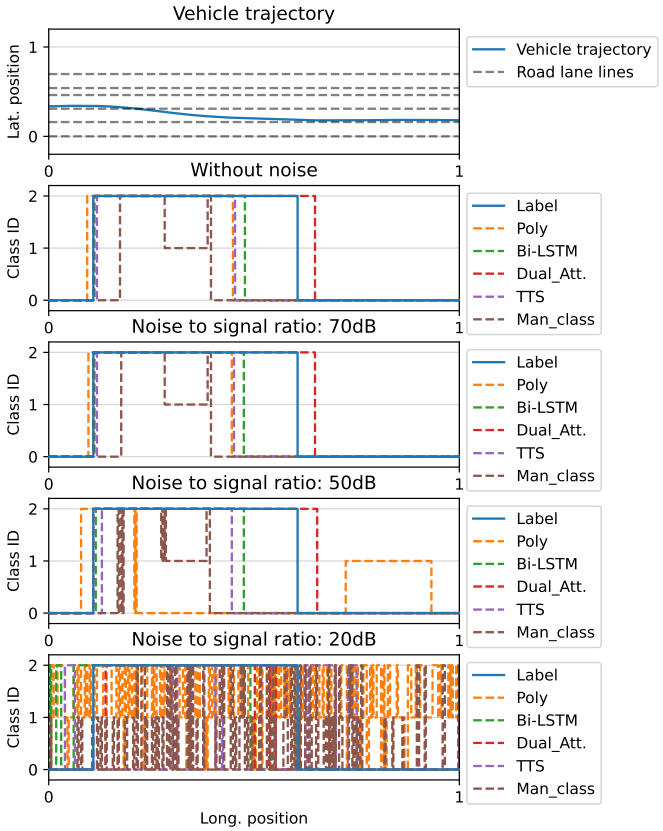


Figure 13: Evaluation example HighD

the *TTS* and *Maneuver-Classification* models. By increasing the noise level to 20 dB, the *Bi-LSTM* model maintained the highest performance across all metrics. Its ability to handle high noise levels is attributed to the convolutional input layer, which provides a robust feature representation. The subsequent Bi-LSTM layers receive more informative inputs, enabling the model to classify data more accurately than the Dual Attention model, despite both architectures incorporating Bi-LSTM layers. The polynomial model, by contrast, performed significantly worse across all metrics except for Center Offset. This indicates that it remains effective at estimating the center position of the lane change but performs poorly on other evaluation criteria. This behavior results from the model's use of polynomial representations for both vehicle trajectories and lane lines, combined with a criterion function that emphasizes curve fitting rather than the longitudinal extent of the lane change. As a consequence, under increased noise, the polynomial method tends to narrow the estimated start and end points of the lane change, bringing them closer into alignment with the lane line crossing point.

Overall, the *Bi-LSTM* model performed best, followed by the Dual Attention model. The *TTS* and *Maneuver-Classification* models took third and fourth place respectively in terms of accuracy, while the polynomial method ranked last. However, when considering other metrics, the polynomial method achieved better results than both the *TTS* and *Maneuver-Classification* models. Additionally, a visual summary of the results for each method and each data subset (HighD, HLCsim, NGSIM) can

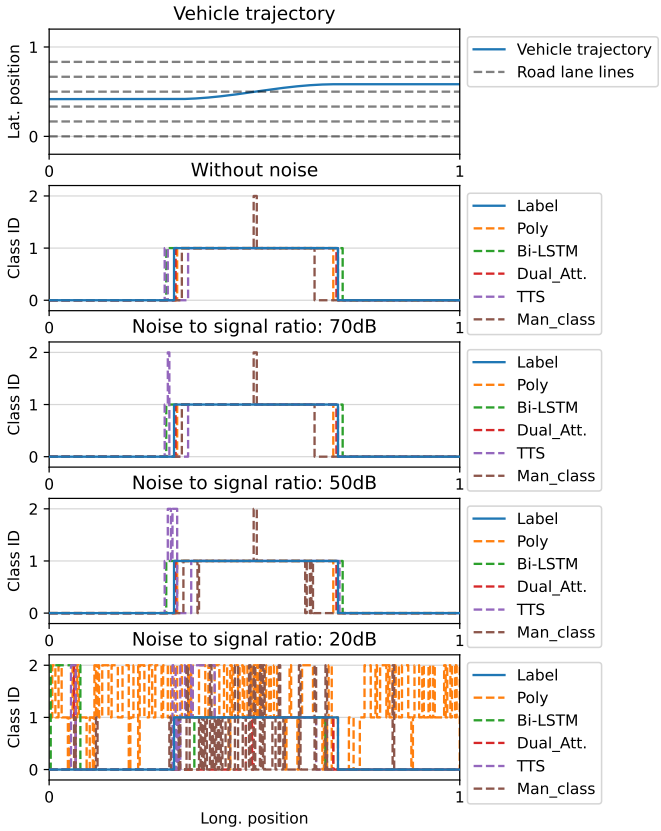


Figure 14: Evaluation example HLCsim

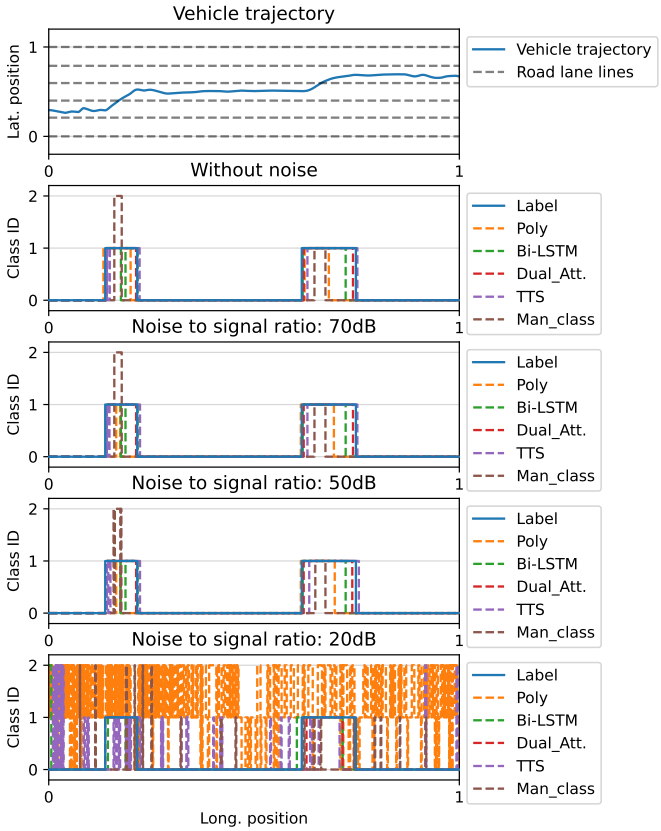


Figure 15: Evaluation example NGSIM

be seen in Figs. 13–15, where each segment is represented by a single entry. Each subplot in Figs. 13–15 represents a different level of noise, ranging from no added noise up to 20 dB.

7. Conclusion

The best overall results were achieved by the *Bi-LSTM* model presented in this work, closely followed by the Dual Attention model [19]. Two other models, *TTS* [21] and *Maneuver-Classification* [20], performed worse and were outperformed by the Polynomial method according to the metrics introduced in this work. The evaluation results suggest that, in broader applications, the neural network models consistently attain superior performance across all parameters compared to the Polynomial method. Notably, the performance disparities between the neural network models and the Polynomial method significantly intensify as the input noise level increases. However, these results were obtained using validation data very similar to the training data, which may not reflect real-world conditions and gives an unfair advantage to the neural network models. The performance of neural networks depends on the characteristics of the input data and may require additional training to adapt to different noise levels. The Polynomial method is parametrizable and more likely to provide continuous and centered labels, making it better suited for low-noise data. The suggested metrics provide additional insights into classification quality. One particularly informative metric was *Center Offset*, as it shows the ability of the Polynomial method to detect the center of the lane change. However, a notable disadvantage is its reliance on labelled data, which limits its application to datasets that include labels.

Possible future work could involve expanding the metric functions to work with unlabeled data and including these metrics in the loss function for neural network training. Additional metrics that simulate scenarios based on classification output could be incorporated. The Polynomial method and its criterion function can be modified to adapt the method for data where results are not usable. These modifications could involve incorporating additional preprocessing steps or modifying criteria or thresholds to handle cases where the method may not produce satisfactory results. One possible modification is to introduce a minimum longitudinal distance requirement for lane changes. This modification results in the Polynomial method considering only lane changes of sufficient length, while filtering out cases where the distance is too short to be considered a valid lane change.

8. References

[1] K. Okamoto, P. Tsiotras, Data-driven human driver lateral control models for developing haptic-shared control advanced driver assist systems, Robotics and Autonomous Systems 114 (2019) 155–171. doi:https://doi.org/10.1016/j.robot.2019.01.020.
URL https://www.sciencedirect.com/science/article/pii/S0921889018302008

[2] D. T. Kanapram, P. Marin-Plaza, L. Marcenaro, D. Martin, A. de la Escalera, C. Regazzoni, Self-awareness in intelligent vehicles: Feature based dynamic bayesian models for abnormality detection, Robotics and Autonomous Systems 134 (2020) 103652. doi:https://doi.org/10.1016/j.robot.2020.103652.
URL https://www.sciencedirect.com/science/article/pii/S0921889020304929

[3] A. Pütz, A. Zlocki, J. Bock, L. Eckstein, System validation of highly automated vehicles with a database of relevant traffic scenarios, 2017.

[4] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, M. Maurer, Defining and substantiating the terms scene, situation, and scenario for automated driving, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 982–988. doi:10.1109/ITSC.2015.164.

[5] E. Andreotti, P. Boyraz, S. Selpi, Mathematical definitions of scene and scenario for analysis of automated driving systems in mixed-traffic simulations, IEEE Transactions on Intelligent Vehicles 6 (2) (2021) 366–375. doi:10.1109/TIV.2020.3031981.

[6] T. Menzel, G. Bagschik, M. Maurer, Scenarios for development, test and validation of automated vehicles (2018). arXiv:1801.08598.

[7] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, M. Maurer, From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment, in: 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE, Paris, France, 2019, pp. 2383–2390. doi:10.1109/IVS.2019.8814099.
URL https://doi.org/10.1109/IVS.2019.8814099

[8] OpenSCENARIO, ASAM - Association for Standardization of Automation and Measuring Systems, [Accessed on June 17, 2023] (2023).
URL {https://www.asam.net/standards/detail/openscenario/}

[9] dSPACE GmbH, dSPACE ASM Scenario Editor, dSPACE GmbH, [Accessed on June 17, 2023] (2023).
URL {https://www.dspace.com/en/pub/home/products/sw/automotive_simulation_models/produkte_asm/modeldesk/traffic_editor.cfm}

[10] H. Beglerovic, T. Schloemicher, S. Metzner, M. Horn, Deep learning applied to scenario classification for lane-keep-assist systems, Applied Sciences 8 (2018) 1. doi:10.3390/app8122590.

[11] S. Zhao, Y. Wu, X. Zhu, B. Zhou, H. Bao, L. Zhang, Research on automatic classification for driving scenarios based on big data and ontology, in: 2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018, pp. 1857–1862. doi:10.1109/CompComm.2018.8780658.

[12] G. Bagschik, T. Menzel, M. Maurer, Ontology-based scene creation for the development of automated vehicles, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, Changshu, China, 2018, pp. 1813–1820. doi:10.1109/IVS.2018.8500632.
URL https://doi.org/10.1109/IVS.2018.8500632

[13] ISO, 26262: Road Vehicles : Functional Safety, Geneva, Switzerland, 2016.

[14] U. Lages, M. Spencer, R. Katz, Automatic scenario generation based on laserscanner reference data and advanced offline processing, in: 2013 IEEE Intelligent Vehicles Symposium (IV), 2013, pp. 153–155. doi:10.1109/IVS.2013.6629463.

[15] R. van der Made, M. Tideman, U. Lages, R. Katz, M. Spencer, Automated generation of virtual driving scenarios from test drive data, in: 24th International Technical Conference on the Enhanced Safety of Vehicles (ESV): Traffic Safety Through Integrated Technologies, National Highway Traffic Safety Administration, NHTSA, Gothenburg, Sweden, 2015, p. 5, paper No. 15-0268.
URL http://www-esv.nhtsa.dot.gov/Proceedings/24/isv7/main.htm

[16] L. Hartjen, R. Philipp, F. Schuldt, F. Howar, B. Friedrich, Classification of driving maneuvers in urban traffic for parametrization of test scenarios, 2019.

[17] H. Zhang, Z. Nan, T. Yang, Y. Liu, N. Zheng, A driving behavior recognition model with Bi-LSTM and Multi-Scale CNN, in: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE, Las Vegas, NV, USA (virtual), 2020, pp. 284–289. doi:10.1109/IV47402.2020.9304772.
URL https://doi.org/10.1109/IV47402.2020.9304772

[18] H. Woo, Y. Ji, H. Kono, Y. Tamura, A. Yamashita, H. Asama, Detection method of lane change intentions in other drivers using hidden markov models, The Abstracts of the international conference on advanced mechatronics : toward evolutionary fusion of IT and mechatronics : ICAM 2015.6 (2015) 253–254. doi:10.1299/jsmeicam.2015.6.253.

[19] F. Ashfaq, R. M. Ghoniem, N. Z. Jhanjhi, N. A. Khan, A. D. Algarni, Using dual attention bilstm to predict vehicle lane changing maneuvers on highway dataset, Systems 11 (4) (2023). doi:10.3390/systems11040196.
URL https://www.mdpi.com/2079-8954/11/4/196

[20] H. Liu, K. Wu, S. Fu, H. Shi, H. Xu, Predictive analysis of vehicular lane changes: An integrated lstm approach, Applied Sciences 13 (18) (2023). doi:10.3390/app131810157.
URL https://www.mdpi.com/2076-3417/13/18/10157

[21] Z. Wan, L. Li, H. Yu, M. Yang, A long short-term memory-based approach for detecting turns and generating road intersections from vehicle trajectories, Sensors 22 (18) (2022). doi:10.3390/s22186997.
URL https://www.mdpi.com/1424-8220/22/18/6997

[22] T. Hülnhagen, I. Dengler, A. Tamke, T. Dang, G. Breuel, Maneuver recognition using probabilistic finite-state machines and fuzzy logic, in: 2010 IEEE Intelligent Vehicles Symposium, 2010, pp. 65–70. doi:10.1109/IVS.2010.5548066.

[23] Yu-Chen Lin, Ha-Ly Thi Nguyen, Cheng-Hsien Wang, Adaptive neuro-fuzzy predictive control for design of adaptive cruise control system, in: 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), 2017, pp. 767–772. doi:10.1109/ICNSC.2017.8000187.

[24] A. Sonka, F. Krauns, R. Henze, F. Küçükay, R. Katz, U. Lages, Dual approach for maneuver classification in vehicle environment data, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 97–102. doi:10.1109/IVS.2017.7995704.

[25] V. Diklić, I. Marković, Lane change classification with neural networks for automated conversion of logical scenarios, in: I. Petrovic, E. Menegatti, I. Marković (Eds.), Intelligent Autonomous Systems 17, Springer Nature Switzerland, Cham, 2023, pp. 778–789.

[26] V. Diklić, Highway lane change from simulation (2023). doi:10.34740/KAGGLE/DSV/5844662.
URL https://www.kaggle.com/dsv/5844662

[27] Products - dSPACE, https://www.dspace.com/en/inc/home/products, (Accessed on 01/26/2021) (2021).

[28] Y. He, B. Cao, C.-Y. Chan, WUT-NGSIM: A high-precision and trustworthy vehicle trajectory dataset (2022). doi:10.21227/hmsb-ka76.
URL https://dx.doi.org/10.21227/hmsb-ka76

[29] R. Krajewski, J. Bock, L. Kloeker, L. Eckstein, The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2118–2125. doi:10.1109/ITSC.2018.8569552.

[30] U. Z. A. Hamid, F. R. A. Zakuan, K. A. Zulkepli, M. Z. Azmi, H. Zamzuri, M. A. A. Rahman, M. A. Zakaria, Autonomous emergency braking system with potential field risk assessment for frontal collision mitigation, in: 2017 IEEE Conference on Systems, Process and Control (ICSPC), 2017, pp. 71–76. doi:10.1109/SPC.2017.8313024.

[31] D. L. Luu, C. Lupu, T. Van Nguyen, Design and simulation implementation for adaptive cruise control systems of vehicles, in: 2019 22nd International Conference on Control Systems and Computer Science (CSCS), 2019, pp. 1–6. doi:10.1109/CSCS.2019.00008.

[32] N. Hashimoto, S. Thompson, Y. Takinami, Evaluation experiments for lane keeping assistance system under several conditions, in: 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018, pp. 442–445. doi:10.1109/CSCI46756.2018.00089.

[33] CarMaker — IPG Automotive — ipg-automotive.com, https://ipg-automotive.com/en/products-solutions/software/carmaker/, [Accessed 05-Jul-2023] (2023).

[34] CarSim Overview — carsim.com, https://www.carsim.com/products/carsim/index.php, [Accessed 05-Jul-2023] (2023).

[35] Simulation Software • Training • Testing • Validation • rFpro • Southampton — rfpro.com, https://rfpro.com/simulation-software/, [Accessed 05-Jul-2023] (2023).

[36] C. Team, CARLA — carla.org, https://carla.org/, [Accessed 05-Jul-2023] (2023).

[37] Detail — asam.net, https://www.asam.net/standards/detail/openscenario/, [Accessed 05-Jul-2023] (2023).

[38] D. Bassermann, About ASAM — asam.net, https://www.asam.net/about-asam/, [Accessed 05-Jul-2023] (2023).

[39] W. Nelson, Continuous-curvature paths for autonomous vehicles, in: Proceedings, 1989 International Conference on Robotics and Automation, 1989, pp. 1260–1264 vol.3. doi:10.1109/ROBOT.1989.100153.

[40] B. Zhou, Y. Wang, G. Yu, X. Wu, A lane-change trajectory model from drivers' vision view, Transportation Research Part C: Emerging Technologies 85 (2017) 609–627. doi:10.1016/j.trc.2017.10.013.

[41] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P. Muller, Deep learning for time series classification: a review, Data Mining and Knowledge Discovery 33 (4) (2019) 917–963. doi:10.1007/s10618-019-00619-1.

[42] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). arXiv:1412.6980.

[43] C. Shorten, T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (1) (2019) 60.

[44] J. Kim, K. Jo, W. Lim, M. Lee, M. Sunwoo, Curvilinear-coordinate-based object and situation assessment for highly automated vehicles, IEEE Transactions on Intelligent Transportation Systems 16 (3) (2015) 1559–1575. doi:10.1109/TITS.2014.2369737.

[45] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, K. Kuhnert, A lane change detection approach using feature ranking with maximized predictive power, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 108–114. doi:10.1109/IVS.2014.6856491.

[46] S. A. Zekany, R. G. Dreslinski, T. F. Wenisch, Classifying ego-vehicle road maneuvers from dashcam video, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, pp. 1204–1210. doi:10.1109/ITSC.2019.8917001.

[47] K. Liang, J. Wang, A. Bhalerao, Lane change classification and prediction with action recognition networks (2022). arXiv:2208.11650.