Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Multi-Modal Two-Wheeler Trajectory Prediction via Representation Learning

Swathy Muthukrishnan

Course of Study:	M.Sc. Information Technology
Examiner:	Dr. Ilche Georgievski
Supervisor:	Karl Ludwig Stolle-M.Sc.
Commenced:	June 9, 2023
Completed:	December 9, 2023

Acknowledgements

I express my heartfelt gratitude to Mr. Karl Stolle Ludwig for granting me the opportunity to delve into this engaging thesis topic and for his unwavering support and guidance throughout this journey. Additionally, I deeply appreciate Dr. Ilche Georgievski for his invaluable guidance and feedback.

I extend my sincere thanks to the esteemed faculty of the Master's program in Information Technology at the University of Stuttgart for their continuous mentorship, which has significantly contributed to my career growth. I am especially grateful to my best friend for his unwavering support during the completion of my thesis.

Lastly, I am profoundly thankful to my family, whose enduring love has been an unceasing source of motivation throughout my academic pursuits. I dedicate this thesis to my father, Muthurkishnan, mother, Pusha, and my beloved late brother, Vasanth Muthukrishnan, whose unwavering support made my Master's journey possible.

Abstract

Powered Two-Wheeler (PTW)s are a class of Vulnerable Road Users (VRU). They are easily susceptible to accidents due to the inherent and unique kinematics of PTWs that have different dynamics while in motion and while stationary. Roll angles of PTW are highly correlated with their lateral dynamics and complement existing safety systems like the Advanced Rider Assistance System (ARAS) and Motorcycle Stability Control (MSC). This thesis work aims to address the safety concerns by contributing to roll angle predictions in a multi-modal fashion using a data-driven approach. Unlike most state-of-the-art research on Multi-Modal Trajectory Prediction (MTP), which used multiple information modalities, the proposed research work centered around using vehicle dynamics data along with a few rider behavior signals. The constraint on the available modalities of data posed a significant challenge, and the proposed method overcomes these challenges by using deep representation learning techniques on the available dataset. Initially, an Autoencoder (AE) was used to reduce the dimensionality of the input data to enhance clustering. To introduce the notion of modality, Gaussian Mixture Model (GMM) was used over the condensed data to identify clusters in an unsupervised approach. The identified clusters were then used to formulate modes, the cornerstones for envisioning a multi-modal approach. The distinct clusters encapsulate data points that exhibit different variations in vehicle dynamics. It was found that clusters with high variations were associated with high lateral dynamic activities such as lane changes, lane merges, and turning. They were distinct from groups exhibiting low dynamics, such as straight riding. The proposed MTP hybrid model, realized using the identified clusters, consists of two significant parts: the Long Short-Term Memory (LSTM) classifier that predicts the future modes and a set of LSTM units, each dedicated to giving roll angle predictions belonging to a particular mode. The MTP model was tested under ideal and realistic conditions. While we have seen promising improvement in roll angle predictions in the ideal situation, there are challenges when predicting outcomes with a noisy classifier. The significant class imbalance in vehicle trajectory information makes it hard to explore ways to add more data without the risk of mislabelling. Variational Auto Encoder (VAE) and other data augmentation techniques help mitigate class imbalance. However, in the context of PTW trajectory data, additional overhead is needed to validate if the generated data aligns well with the unique physics governing the stability of PTWs. The roll angle outcomes tested under the two conditions showed that with limited data modality, MTPs were feasible, but accurate information about future conditions was crucial for improving prediction accuracy. Our findings set the stage for future improvements in PTW safety systems, emphasizing the importance of having precise information about future conditions.

Kurzfassung

Powered Two-Wheeler (PTW)s sind eine Art von gefährdeten Straßenbenutzern (VRU) und sind aufgrund der inhärenten und einzigartigen Kinematik von PTWs, die unterschiedliche Dynamiken sowohl in Bewegung als auch im Stillstand aufweisen, leicht an Unfälle zu riskieren. Rollwinkel von PTWs korrelieren stark mit ihrer lateralen Dynamik und ergänzen bestehende Sicherheitssysteme wie das Advanced Rider Assistance System (ARAS) und die Motorrad-Stabilitätskontrolle (MSC). Diese Dissertation zielt darauf ab, die Sicherheitsbedenken zu adressieren, indem sie zu Rollwinkelvorhersagen auf eine multimodale Weise mittels eines datengetriebenen Ansatzes beiträgt. Im Gegensatz zu den meisten Forschungen zum Multi-Modal Trajectory Prediction (MTP), die mehrere Modalitäten von Informationen verwendet haben, konzentrierte sich die vorgeschlagene Forschungsarbeit darauf, Fahrzeugdynamikdaten zusammen mit einigen Signalen zum Fahrerverhalten zu verwenden. Die Einschränkung der verfügbaren Modalitäten von Daten stellte eine bedeutende Herausforderung dar, und die vorgeschlagene Methode überwindet diese Herausforderungen durch den Einsatz von Techniken des tiefen Darstellungs-Lernens auf dem vorhandenen Datensatz. Zunächst wurde ein Autoencoder (AE) verwendet, um die Dimensionalität der Eingabedaten zur Verbesserung der Clusterbildung zu reduzieren. Zur Einführung des Begriffs der Modalität wurde über die kondensierten Daten ein Gaussian Mixture Model (GMM) in einem unüberwachten Ansatz verwendet, um Cluster zu identifizieren. Die identifizierten Cluster wurden dann verwendet, um Modi zu formulieren, die die Eckpfeiler für die Vorstellung eines multimodalen Ansatzes darstellen. Die verschiedenen Cluster umfassen Datenpunkte, die verschiedene Variationen in der Fahrzeugdynamik aufweisen. Es wurde festgestellt, dass Cluster mit hohen Variationen mit Aktivitäten hoher lateraler Dynamik wie Spurwechsel, Fahrstreifenwechsel und Abbiegen verbunden sind und sehr verschieden von Gruppen mit geringer Dynamik wie geradem Fahren sind. Das vorgeschlagene MTP-Hybridmodell, das mit den identifizierten Clustern realisiert wurde, besteht aus zwei wesentlichen Teilen: dem Long Short-Term Memory (LSTM) Klassifizierer, der die zukünftigen Modi vorhersagt, und einer Reihe von LSTM-Einheiten, die jeweils Rollwinkelvorhersagen für einen bestimmten Modus liefern. Das MTP-Modell wurde unter idealen und realistischen Bedingungen getestet. Während vielversprechende Verbesserungen bei der Rollwinkelvorhersage in idealen Situationen festgestellt wurden, gibt es Herausforderungen bei der Vorhersage von Ergebnissen mit einem rauschenden Klassifizierer. Das signifikante Klassenungleichgewicht bei den Fahrzeugtrajektoriedaten erschwert die Erforschung von Möglichkeiten, weitere Daten hinzuzufügen, ohne das Risiko einer Fehlkennzeichnung einzugehen. Variational Auto Encoder (VAE) und andere Techniken zur Datenanreicherung tragen dazu bei, das Klassenungleichgewicht zu mildern, aber im Kontext von PTW-Trajektoriendaten ist zusätzlicher Aufwand erforderlich, um zu validieren, ob die generierten Daten gut mit der einzigartigen Physik, die die Stabilität von PTWs regelt, übereinstimmen. Die Rollwinkelergebnisse, die unter den beiden Bedingungen getestet wurden, zeigen, dass mit begrenzter Datenmodalität MTPs machbar sind, aber genaue Informationen über zukünftige Bedingungen sind entscheidend für die Verbesserung der Vorhersagegenauigkeit. Dies bildet die Grundlage für zukünftige Verbesserungen in den PTW-Sicherheitssystemen und betont die Bedeutung präziser Informationen über zukünftige Bedingungen.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ABS Anti-lock Braking System. 17

AE Autoencoder. 4

AI Artificial Intelligence. 29

ANN Artificial Neural Network. 20

ARAS Advanced Rider Assistance System. 4

BDLSTM Bidirectional LSTM. 23

BDLSTMAE BDLSTM AE. 37

BERT Bidirectional Encoder Representations from Transformer. 18

BPTT Backpropagation Through Time. 22

CIA Controlled Interpolation Algorithm. 66

CNN Convolutional Neural Network. 22

DAE Deep Linear AE. 9

DBN Dynamic Bayesian Network. 28

DL Deep Learning. 19

GAN Generative Adversarial Network. 29

GMM Gaussian Mixture Model. 4

GPS Global Positioning System. 17

IMU Inertial Measurement Unit. 17

IPL Input Sequence Length. 37

KDE Kernel Density Estimation. 46

KNN K-Nearest Neighbor. 29

LC1 LSTM Classifier1. 76

LC2 LSTM Classifier2. 76

LC3 LSTM Classifier3. 76

LIDAR Light Detection and Ranging. 29

LS Latent Space. 17

LSD LS Dimension. 37

LSTM Long Short-Term Memory. 4

LSTMAE LSTM AE. 37

ML Machine Learning. 28

MLP multilayer perceptron. 20

MSC Motorcycle Stability Control. 4

MSE Mean Squared Error. 37

MTP Multi-Modal Trajectory Prediction. 4

PC principal component. 45

PCA Principal Component Analysis). 45

PDF Probability density function. 46

PTW Powered Two-Wheeler. 4

ReLu Rectified Linear Unit. 25

RMSE Root Mean Square Error. 20

RNN Recurrent Neural Network. 22

SD Standard Deviation. 53

SSR Sum of squares. 40

SSTO total sum of squares. 40

TF Transformer. 17

UTM Universal Transverse Mercator. 28

VAE Variational Auto Encoder. 4

1 Introduction

The European motorcycle market saw an increase in PTW usage in the first half of 2023 by 11.3%, and the German market, in particular, showed an increase of 10.1% [Ord]. With this evident rise in PTWs, there is an inherent demand for rider safety, and this has been a topic of concern even in the past. An EU accident report from 2018 indicates that VRUs accounted for about 47% of all the European road fatalities that occurred in 2016. Specifically, 17% of these fatalities were caused by the PTWs and 8% by the cyclists [BMT+21]. This safety concern is further underscored in the Safer Wheels study, which illustrates that 25% of the 500 PTW-bicycle accidents in six European countries were caused by the rider losing control when navigating curves. Hence, the increasing use of PTWs and their associated accidents calls for predictive trajectory models.

Roll angles of PTWs are highly correlated with PTW dynamics and stability, and their estimation helps supplement existing active safety systems such as ARAS, Anti-lock Braking System (ABS), and MSC [HBCY13]. Hence, by predicting PTW trajectories and implementing proactive safety measures, we can effectively address the challenges posed by the surge in PTW usage, thus minimizing potential risks and fostering a safer environment for all road users. However, existing safety initiatives primarily focus on four-wheeled vehicles, and there is a significant gap in research dedicated to four-wheelers and PTWs. This gap is further widened by the distinctive challenges arising from the physics behind PTWs that are significantly different from the former. Therefore, models that anticipate and mitigate risks associated with PTW maneuvers address overlooked aspects of two-wheeler dynamics and bridge a significant gap in road safety.

The proposed methodology focuses on harnessing the power of MTP to predict roll angles for PTW based solely on vehicle dynamics and select rider behavior data. Hence, the original dataset introduced in section 2.1.1 was the only data used, and no additional sensory information was collected. Furthermore, the Global Positioning System (GPS) information and context information surrounding the ego vehicle were deliberately avoided, and the reason for this is two-fold: (1) A significant portion of the ride information encapsulated in the original dataset was recorded on the rural roads in southern parts of Germany. Hence, there was an inherent risk of GPS information loss and a lack of information about traffic regulations. (2) The proposed methodology targets motorcycles, meaning we could access only the channels available in the Inertial Measurement Unit (IMU) and a few onboard sensors. Since MTP is more feasible with distinct information groups, a significant portion of the thesis employed deep representation learning to identify interesting patterns from the existing dataset. Initially, we aimed to discover a reliable encoding for the vehicle dynamics, effectively capturing their Latent Space (LS) representation. Subsequently, a suitable unsupervised clustering algorithm, such as mixture models, was used on the LS encoding to identify clusters. Finally, employing the clusters obtained from the chosen clustering algorithm, our MTP framework, predominantly relying on LSTM networks, was tested for MTP. This choice of neural networks was made for two primary reasons: alignment with the existing baseline established on LSTM and a preference for a pragmatic approach over complex models like Transformer (TF), Bidirectional Encoder Representations from Transformer (BERT), or temporal convolutions. By iteratively refining this methodological pipeline, we aimed to achieve precise and reliable predictions for improved safety and predictive accuracy. This methodology was executed in successive stages to ensure robustness and accuracy in predictions.

1.1 Goal of the Thesis

Following the intricate challenges associated with data, the subsequent steps for realizing MTP involved formulating the key research questions answered in this study. The articulated research questions are presented below:

RQ1 Are there cluster groups within vehicle dynamics data and rider behavior signals?

RQ2 Can the cluster groups be used to formulate consistent modes across the prediction horizon?

RQ3 Is it feasible to realize MTP with just the identified modes?

RQ4 Can the proposed MTP model improve roll angle predictions compared to the baseline benchmark?

1.2 Thesis Overview

The remainder of the thesis work is structured as follows: Chapter 2 begins with an overview of the baseline model and dataset employed at Bosch, followed by an introduction to the various neural networks used in this study. Chapter 3 offers a comprehensive review of relevant literature. Subsequently, Chapter 4 presents the proposed methodology for implementing MTP. Given the sequential nature of our approach, Chapters 5 and 6 delve into the experimental setup, outcomes, and insights derived from representation learning using AE and the experimentation and results obtained from utilizing GMM for clustering, respectively. Building upon the results of these foundational components, which are a prerequisite for designing the hybrid model for MTP, Chapter 7 elucidates the structured experimental settings for the MTP hybrid model. The culmination of the MTP experiments is detailed in Chapter 8, outlining the final results. Finally, Chapter 9 encapsulates this thesis's comprehensive conclusion and outlook.

2 Base Line Model and Background Information

2.1 Existing Work

The existing work at Bosch revolves around realizing unimodal roll angle trajectory predictions for PTWs which gives more insight into how PTW maneuvers undergo cornering. Instead of focusing on forward velocity, the aim was to understand the lateral dynamics by forecasting the future roll angles of PTWs [SWS23]. This endeavor sought to decipher the riders' intentions as they cornered, considering the time delay between their actions and the PTW's lateral movements. The approach was rooted in harnessing the intricate patterns in rider data, utilizing Deep Learning (DL) techniques to accurately predict cornering behavior.

2.1.1 Dataset

The dataset used in this study encompassed a comprehensive collection of riding data sourced from a KTM 1290 Super Adventure motorcycle, which was equipped with a few additional sensors. These sensors captured a wide spectrum of essential information, ranging from intricate vehicle dynamics to detailed rider behavior. This dataset was a rich repository of diverse riding scenarios, encompassing various terrains and riding styles prevalent in Southwest Germany. It was collected for more than 65 hours of riding, involving 21 riders of differing experience levels and riding styles. The dataset's significance lay not just in its volume but in its focus on real-world riding scenarios, particularly emphasizing rural road environments. In this pivotal setting, many single-riding accidents occur.

The uniqueness of this dataset lies in its exclusion of environmental sensing information, focusing solely on internal PTW signals, including distinctive measurements such as roll angle, roll rate, steering torque recorded from the Internal Measurement Unit(IMU), and rider body movements captured by a camera-based system. By purposefully excluding external environmental data, the study accentuated the importance of understanding and predicting rider intentions solely based on the PTW's internal dynamics. This emphasis on real-world riding data, alongside the careful selection of pertinent features, formed the backbone of the study's predictive model, enabling a comprehensive analysis and accurate prediction of rider behavior during cornering. The total number of crucial features identified by this approach amounts to 16, and the same set of features is used throughout this thesis work.

2.1.2 Base Line Model

The DL model employed LSTM layers followed by a multilayer perceptron (MLP) to delve into the predictive capabilities based solely on internal PTW signals.

The study evaluated the model's performance using metrics like Root Mean Square Error (RMSE)to compare the DL model's predictions against a basic constant roll angle heuristic. The findings were striking, demonstrating a significant improvement in predictive accuracy with the DL model over simplistic heuristic methods. The Base Line Model exhibited an overall RMSE of 7.7668 deg for roll angle predictions over a prediction horizon of 4s and took in an input sequence of 1.6s. Additionally, an insightful ablation study highlighted the criticality of non-common measurement signals—specifically, the steering system and rider behavior feature—in augmenting the predictive prowess of the DL model for discerning cornering behavior. This research showcased promising prospects in foreseeing rider intentions related to lateral dynamics, offering a sophisticated leap beyond conventional approaches.

2.2 Relevant Neural Network Architectures

Throughout this thesis work, variations of the basic AE, LSTM neural networks were used, and the following section gives a brief introduction to the relevant architectures.

2.2.1 AE

AE, a fundamental construct in Artificial Neural Network (ANN), is primarily used in the domain of unsupervised learning, where they are employed for tasks including data reconstruction, dimensionality reduction, anomaly detection, and related applications[ZZCH18]. A vanilla AE comprises two essential neural network components: an encoder and a decoder. The encoding function, $f(x)$, effectively learns the correlation amongst the input features and maps the input data from the data space to LS (reduced dimensions). Conversely, the decoding function, $g(z)$, learns to reconstruct the input data from the LS. In contemporary AEs, the encoding and decoding functions are often stochastic, ensuring that the reconstruction, x', approximates the original input, x, rather than merely duplicating it.

Typically, an AE model's parameters, represented as $\theta \in \{W, b\}$ for encoding and $\theta' \in \{W', b'\}$ for decoding, are optimized through the minimization of a suitable cost function over a training dataset [MMCS11]. For the specific use case of dimensionality reduction, a bottleneck AE architecture is often used to obtain low-dimensional representations from the original data, facilitating feature space reduction. A BN architecture (see Fig. 2.1) consists of a bottleneck layer, primarily a linear layer with fewer neurons than the linear layers found in the encoder and decoder network. This reduction in the number of neurons motivates the model to learn a compact representation of crucial information for reconstructing the data instances efficiently. By minimizing the overall reconstruction error (cost function), the retained information is condensed to be as relevant as possible to most instances, which is particularly useful for dimensionality reduction. As a result,

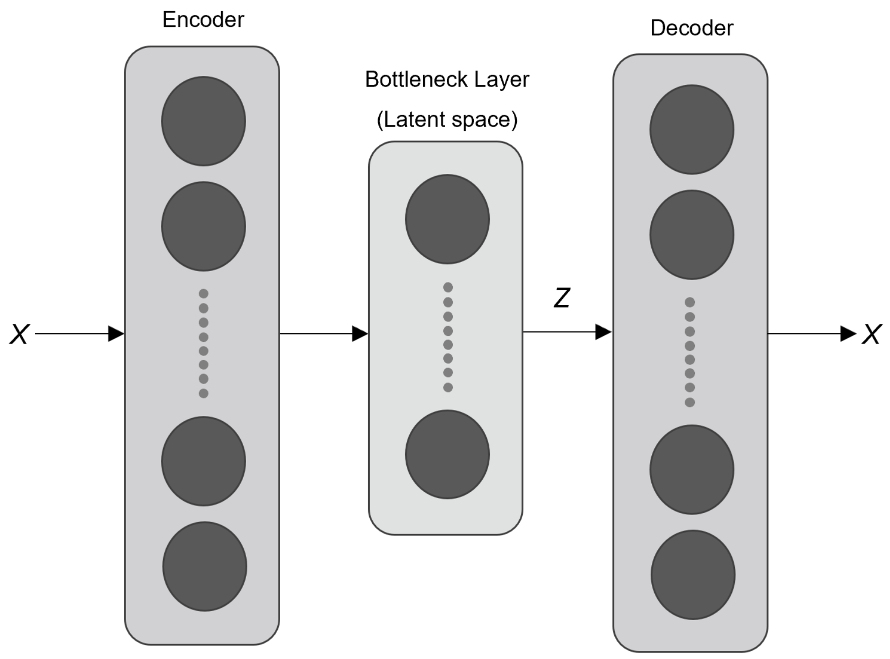


Figure 2.1: X represents the input from the data space. In this simple network, both the encoder and the decoder have a single linear layer with more neurons (n) than the number of neurons in the linear bottleneck layer (m). The output from the bottleneck layer Z provides a reduced representation of the input information passed to the network.

data instances that deviate from the norm may not be well reconstructed, effectively aiding in forming a reduced LS [PSCH22].

If $X \in \{T_n\}$ represents the time samples or data points from the training dataset T and $Z \in \{N_m\}$ represents the features from the reduced LS, then the AE illustrated in Fig. 2.1 can be modeled by the equation 2.1. The function f represents the encoder and the bottleneck layer. Hence, for the network illustrated in Fig. 2.1, the encoder block comprises two linear layers: the last layer has m number of neurons, and the first layer has n number of neurons. Likewise, the function g represents the decoder block, a single linear layer. Θ and Θ' represent the trainable hyper-parameters of the encoder and decoder, respectively. W represents the weights and, b represents the bias of the encoder block, W' and b' represent the weights and biases of the decoder block.

$$\begin{aligned} Z &= f(\theta = \{W, b\}, X) \\ X' &= g\,(\theta' = \{W', b'\}) \end{aligned} \tag{2.1}$$

2.2.2 LSTM

Recurrent Neural Network (RNN) constitute a class of ANNs tailored for processing sequential or time series data. What sets RNNs apart from conventional feedforward and Convolutional Neural Network (CNN) is their intrinsic ability to maintain an internal state or memory at each time step. This capacity allows them to store and utilize information from prior time steps, thereby influencing both the current input and output, a crucial feature for understanding and predicting sequential data. RNNs achieve this by employing feedback loops in their architecture, which enable them to share parameters consistently across all layers and time steps. These shared parameters are updated through the Backpropagation Through Time (BPTT) algorithm, a specialized variant of the standard backpropagation. BPTT unfolds the network over time, ensuring the effective learning of temporal dependencies within the data. However, despite their advantages, RNNs are prone to the issues of gradient exploding and vanishing, which, in practice, limit their ability to effectively incorporate information from the past, typically extending to approximately five to ten steps [SM19]. This limitation led to variants, such as the LSTM.

LSTMs are characterized by a distinctive gating mechanism that enables them to retain information across extended sequences effectively. These gating mechanisms play a critical role in determining what information should be preserved and what should be discarded [AL18][CMC19]. Like RNNs, LSTM networks maintain an internal state or memory at each time step. The gating mechanism is instrumental in continuously maintaining and updating these states, ensuring that they are filled with the most pertinent and essential information. The gates within an LSTM comprise a feedforward layer and a subsequent sigmoid activation function and ultimately involve a pointwise multiplication with the target layer subject to gating [CMC19]. They are not separate neural networks but rather integral components of the LSTM architecture. The information that the LSTM unit maintains at each time step is called the 'context' (sometimes also called the memory or cell state).

The internal structure of an LSTM unit with the three essential gates is shown in Fig. 2.2, and this structure is based on the LSTM unit implemented in the Pytorch framework [PGM+19].

$$
\begin{aligned}
\mathbf{f}_t &= \sigma\left(W_{if}x_t + b_{ik}x_t + W_{bt}h_{t-1} + b_{bt}h_{t-1}\right)\\
i_t &= \sigma\left(W_{ii}x_t + b_{ij}x_t + W_{bi}h_{t-1} + b_{bi}h_{t-1}\right)\\
g_t &= \mathrm{Tanh}\left(W_{ig}x_t + b_{ig}x_t + W_{bs}h_{t-1} + b_{bs}h_{t-1}\right)\\
a_t &= \sigma\left(W_{io}x_t + b_{io}x_t + W_{ho}h_{t-1} + b_{ba}h_{t-1}\right)\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t\\
h_t &= a_t \odot \mathrm{Tanh}\left(c_t\right)
\end{aligned}
\tag{2.2}
$$

From the illustrated cell structure (Fig. 2.2), x_t represents the current input at time t. H_{t-1}, b_t, c_{t-1} and c_t denote the previous hidden state, current hidden state, previous cell state, and the current cell state, respectively. W_{ab} and b_{ab} represent the weights and biases associated with the incoming layer 'a' and the corresponding LSTM gating mechanism 'b'. To be more precise, a represents the dimensions of the layer associated with the incoming layer, and b represents the dimensions of the layer associated with the gating mechanism $a \in \{i, h\}$ (i – current input at time t, h – previous hidden state), and $b \in \{f, g, i\}$ (f –forget gate at time t, g –candidate cell state at t, i – input gate at t).

If we consider input at time t, the hidden layers of the previous time step, $t - 1$, contain valuable information about all the time steps from the past till $t - 1$. This information is the combination of the past inputs and relevant context needed for the network to make predictions or decisions at the current time step 't'. Now, as the name suggests, the function of the forget gate is to compute what information from the past context (at time $t - 1$) needs to be eliminated, given a new input at time t. This functionality of the forget gate is described in the set of equations 2.2 [PGM+19]. The sigmoid activation function outputs a value between zero and one. Hence, if the forget gate ft gives an output of 0, there is nothing important to remember from the context, i.e., forget everything and a value of 1 indicates that the LSTM has to retain the whole context. The input gate, on the other hand, assesses the significance of the incoming information; it signifies high relevance and a value of zero implies no relevance. O_t computes the significant information to store in the current hidden state (which is then used for context selection in the upcoming iterations). f_t acts as a weighing factor to select the relevant information from the previous cell state, and this is done by element-wise vector multiplication (Hadamard product). After the relevant previous information is scaled from c_{t-1}, it is then added to the candidate cell state. The candidate cell state is the Hadamard product of i_t and g_t. g_t produces a value between -1 and 1. Negative values subtract the corresponding elements from the input gate, acting as a filter to decide which information will be updated to the current context. The h_t also gets calculated similarly and is produced by scaling the value of the output gate by the current context.

Bidirectional LSTM: An expansion of the conventional LSTM architecture, Bidirectional LSTM (BDLSTM) networks is made to extract information from sequential input in both forward and backward directions. BDLSTM networks are an extension of the traditional LSTM architecture, designed to capture information from sequential data in both forward and backward directions. The fundamental distinction between BDLSTM and LSTM lies in utilizing two separate hidden layers for this bidirectional analysis. In BDLSTMs, two distinct hidden layers are employed to process the input sequence simultaneously in both forward and backward directions. This bidirectional approach offers a comprehensive perspective on the sequence data, enabling the network to consider the past and future context. This dual analysis is achieved by linking the two hidden layers to a common output layer, allowing them to collaborate and contribute to the final predictions. Eliminating the one-step truncation in bidirectional LSTM is a significant change from the original LSTM design. BDLSTM improves its training procedure by performing a full error gradient calculation instead. In order to provide more efficient learning and prediction on sequential data, the full error gradient technique simplifies the building of BDLSTM models and aligns them with the BPTT standards [CKPW20]. Fig. 2.3 illustrates the processing of an input sequence in both forward and backward orientations to help visualize how BDLSTMs work. The forward and backward LSTMs' outputs are concatenated to create the output predictions produced by BDLSTMs, which provide a thorough depiction of the sequence's dependencies and context.

2.2.3 Deep AE

Deep AE extend the traditional AE model by incorporating multiple hidden layers in both the encoder and decoder. They are designed to acquire hierarchical data representations, proficiently encapsulating intricate patterns and high-level features [PWW+23]. The 'hourglass architecture' employs encoder and decoder components with multiple linear layers, creating a symmetrical

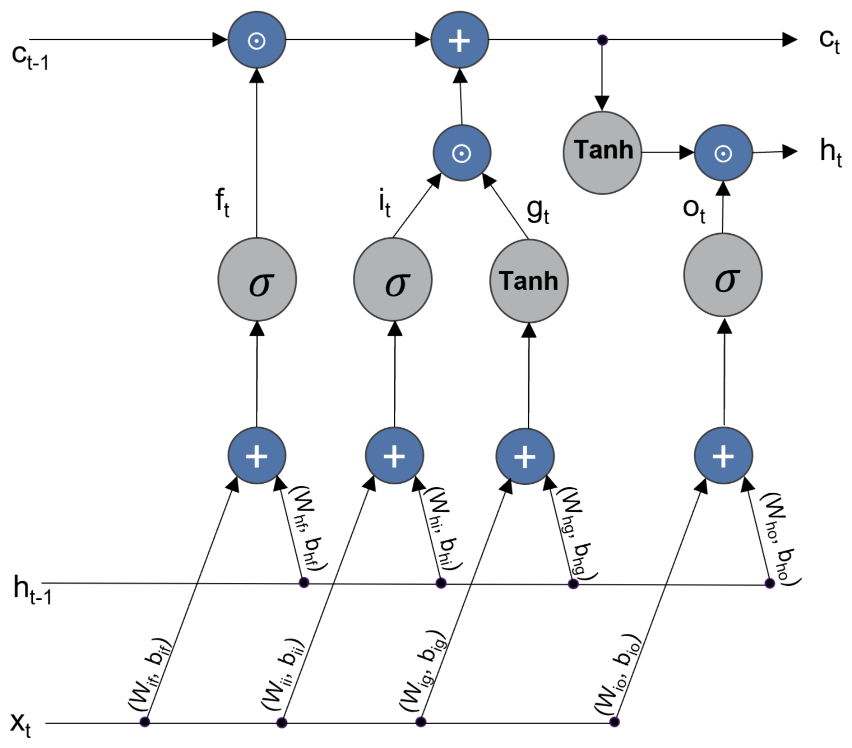


Figure 2.2: The previous arrows shown in the figure are a simple representation of a linear layer, '.' represents the necessary linear transformation needed for the current input and previous hidden states. The Hadamard product is illustrated as $\odot$ and is used to denote element-wise vector multiplication. The sigmoid and tanh activation functions are represented with Σ and $Tanh$, respectively.

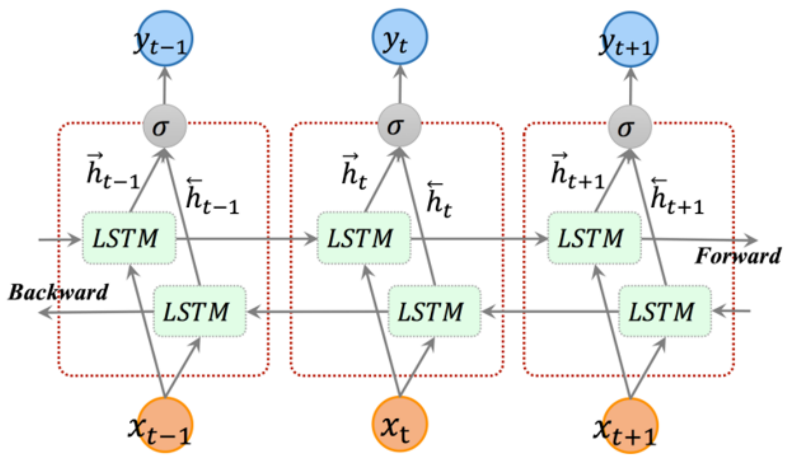


Figure 2.3: Simplified Structure of a BDLSTM [CKPW20]. The time series input sequence is processed in both directions

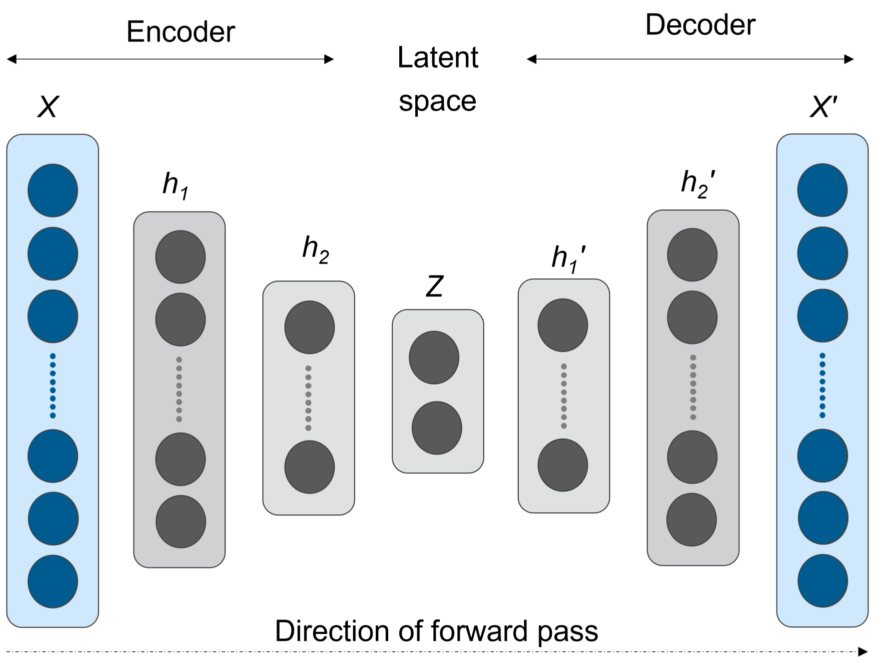


Figure 2.4: X represents the activation for the input layer of the encoder block. X' represents the activation of the output layer in the decoder block. The actuation for the hidden layers at the encoder and the decoder block are represented as h_n and h'_m, respectively, where n and m denote the number of hidden layers in the encoder and decoder block respectively. Z denotes the activation for the LS

hourglass shape. Combining this construct with non-linear activation functions like Tanh and Rectified Linear Unit (ReLu) promotes complex hierarchical feature learning. Time series data often exhibits complex patterns at various scales and levels of abstraction. The hourglass AE comprehends these patterns, traversing from fine-grained details to high-level features while progressively reducing data dimensionality. This configuration simplifies data representation and preserves essential information. Deep AEs adopting an hourglass design serve as proficient denoising filters, reducing dimensionality and noise across encoding layers, followed by data reconstruction [PWW+23]. This network design is particularly valuable for handling noisy or unrefined datasets. Hourglass architectures are well-suited for capturing sequential nuances in time series data, enhancing the understanding of dynamic data trends. Fig. 2.4 illustrates the basic structure of a simple Deep AE with two hidden layers on the encoder and decoder block. The activation of the input and the output layer of the Deep AE can be denoted with X and X', and the activations for the hidden layers at the encoder and the decoder side can be represented as h and h'. If there are 'i' number of hidden layers and if sigma denotes a non-linear activation function and W_n, b_n represents the weights and biases associated with hidden layers of the encoder, and if W'_m, b'_m represents the weights and biases associated with the hidden layers of decoder block, then the activations of the hidden layers can be formulated with equation 2.3. The number of hidden layers at the encoder and decoder block are represented by n and m, respectively.

$$
\begin{aligned}
h_1 &= \sigma\left(W_1^*(X) + b_1\right) \\
h_i &= \sigma\left(W_i^*\left(h_{i-1}\right) + b_i\right), i = \{2, \ldots .n\} \\
h_1' &= \sigma\left(W_i'^*(Z) + b_1'\right) \\
b_i' &= \sigma\left(W_i'^*\left(h_{1-1}\right)' + b_i'\right), j = \{2, \ldots m\}
\end{aligned}
\tag{2.3}
$$

2.2.4 LSTMs with AE

An LSTM AE is an AE architecture that leverages LSTM units. When LSTMs are integrated into an AE, the resultant model becomes adept at capturing and reconstructing sequential patterns within the data. The following section will delve into the fundamentals of a single LSTM unit and a Bidirectional LSTM network, exploring their mechanisms for incorporating temporal patterns in time series data.

3 State of the Art

Harnessing the power of data-driven approaches such as DL offers a transformative potential to revolutionize road safety strategies. These methods, grounded in collecting, analyzing, and interpreting real-world data, are vital to unraveling intricate patterns and uncovering insights that can pave the way for more effective predictive trajectory models and proactive safety measures. Numerous research endeavors have delved into this application scope, exploring innovative methodologies that enhance the prediction of vehicle trajectories.

In the context of trajectory prediction, DL models, notably LSTM networks and Transformers (TF), have emerged as compelling tools. While these models hold promise for predicting trajectory behaviors, it is essential to acknowledge that their application has predominantly centered around four-wheeler vehicles, leaving a notable gap in addressing the specific dynamics and challenges faced while addressing trajectory prediction for PTWs. Moreover, it is worth noting that the successful implementation of such DL methods often requires integrating multimodal sensory information. These methods frequently rely on a combination of visual, spatial, temporal, and social (vehicle interaction) data to accurately predict trajectories and account for complex real-world scenarios. For instance, a recent study [AL18] explored utilizing LSTM networks while addressing the constant velocity challenge commonly encountered on Highways. The core hypothesis was that incorporating localized information from surrounding vehicles could accurately predict a target vehicle's future behavior. [OMCG17] demonstrates using a Bidirectional Recurrent Neural Network (BRNN) for driver action prediction. The central idea behind the research is to formulate the prediction task as an anomaly detection problem. BRNN achieves this by modeling the feature sequence's transition between the non-action and action states. The proposed method can accurately predict key driver actions, including acceleration, braking, lane change, and turning, at durations of 5 before the driver executes the action.

The research in [QFP+21] demonstrates the first-ever use of TF models in the specific context of vehicle trajectory. While the approach seems promising, the experimentation was performed on rich datasets such as inD, round, high, and INTERACTION, which questions the feasibility of such an approach under realistic data acquisition settings. In a similar context, another research was proposed on using TF to predict vehicle trajectories in urban scenarios while using Augmented Information Transformers and Bidirectional Encoder Representations from Transformers (BERT) to predict human trajectories [GHCG21]. The TF utilizes a modular architecture with encoder and decoder layers and operates autoregressively. Its attention mechanism compares query entries with keys to weigh values, capturing complex sequence patterns. [QFP+21], [GHCG21] demonstrates the predictive capacities of simple transformers, drawing attention to the mechanism behind more complex and more protracted sequence transformer variants such as Block-Recurrent Transformers [HSW+22]and transformer-XL [DYY+19]. These models, though powerful, are more suitable for Natural Language Processing and would need extensive reformulation to make them suitable for predicting PTW trajectories. Another transformer-based approach would be to use Temporal Fusion transformers for PTW trajectory prediction. However, this architecture depends on the availability of static covariates in the dataset, which is often different in realistic data acquisition, where the only reliable data is the past and current vehicle dynamics.

The use of maneuver or lane change information, in addition to the vehicle's track history, has shown significant improvement in trajectory prediction problems [Abd20] - [GSC+21]. The Trajectory Prediction system described in [Abd20] comprises two main components: a Manoeuvre Classification Module and a Trajectory Classification Module. Manoeuvre classification uses a neural network (NN), while trajectory prediction uses an LSTM network. The classification of lane changes is learned using an Artificial Neural Network (ANN), with each lane segment's data sampled at a rate of 100 milliseconds over a 5-second interval. The empirical findings indicate that lane changes occur approximately 4 seconds prior to the actual time of the change. [PMLL20] presents a method that leverages multimodal driving context and Machine Learning (ML) techniques to learn latent features for driving maneuver detection, which involves extracting curvature features from GPS data and vehicle signals. GPS points are transformed into Universal Transverse Mercator (UTM) coordinates, and quadratic polynomial fitting is used to fit UTM points. Notably, turn events (left and right) and suitable lane change events exhibit reduced spatial variation. The authors of [HBCY13] Introduce a model-based approach focusing on accurate vehicle trajectory prediction. It presents a prediction method utilizing the Constant Yaw Rate and Acceleration (CYRA) motion model, demonstrating high accuracy for short-term predictions. Additionally, the paper proposes maneuver recognition for longer-term predictions, employing a deterministic and efficient method based solely on kinematic measurements and road geometry detection. [LDL+19] The Dynamic Bayesian Network (DBN) for maneuver prediction adopts a three-layered structure, encompassing causal evidence, maneuver, and diagnostic evidence, leveraging road structure, vehicle interactions, and physics-based features. It learns feature states ' probabilities through maximum likelihood estimation and probabilistic inference and predicts future maneuvers based on acquired probabilities, handling uncertainties via soft evidence. Structured around rational driving behaviors, the DBN integrates multiple predictive features, discretizes continuous variables, and emphasizes inter-causal reasoning to model the decision-making process involved in driving maneuvers. [GSC+21] the paper proposes a cyclist trajectory prediction method utilizing a DBN to infer intention distributions at intersections and LSTM with encoder-decoder for accurate trajectory prediction, aiming to improve prediction accuracy and speed in real traffic scenarios. However, these approaches rely on GPS information or surrounding vehicle context, sometimes both. The work proposed in [ZZTL22] uses a Temporal Convolutional Network for predicting lane changes. It takes in the steering angle along with continuous records of the longitudinal and lateral positions of the ego vehicle in two dimensions. Even though we can convert lateral and longitudinal accelerations into positional information, the proposed method was evaluated only on one simulated condition, a two-lane freeway, with a recurring scenario involving a lead vehicle. As the studies above demonstrate, integrating maneuver information into trajectory prediction and driving behavior detection methodologies showcases a compelling avenue for enhancing prediction accuracy and understanding vehicle dynamics.

Further studies take incorporating maneuver information to the next level by employing representation learning to identify modes based on different maneuvers and then implement MTP with the identified modes. The authors of [ZJP+21] propose a representation learning method using a TF base network to produce the encodings-based input reconstruction with reduced error as an objective.

[LHH+18] It takes a similar approach but uses AE to handle the representation learning tasks but uses LSTM layers in the encoder and decoder parts. Instead of learning to reconstruct the input sequence, it forecasts the output sequences. This way, the representation also contains information that's needed for forecasting. The described framework, Multiple Futures Predictor (MFP)[TS19], is designed to predict future motions of all relevant agents in complex and dynamic environments, which includes capturing interactions among multiple agents in a scene. MFP employs latent variables that serve the purpose of learning diverse and meaningful modes of behavior or potential trajectories without explicit annotations. These latent variables are instrumental in capturing multimodal predictions and modeling interactions among agents during future trajectory prediction. Similar to [TS19], Generative Adversarial Network (GAN)-based approach outlined in [HMD+20] harnesses a latent space to generate diverse and realistic trajectories for vehicle prediction tasks. The latent space, acting as a low-dimensional feature representation, encapsulates high-level semantic information about various driving maneuvers. The model generates synthetic trajectory samples that reflect different semantic categories or behaviors encoded in this latent space. However, the latent sampling approach described in [TS19] has access to social interactions of the ego vehicle with the surrounding vehicle. Similarly, The latent space sampling in[HMD+20] was possible because the annotations needed at the time of sampling were available since the research was done over a publicly available Argoverse forecasting dataset, which has a more modality of information in the form of Camera input, Light Detection and Ranging (LIDAR) point clouds, GPS information and a few other to give annotations for surrounding information. The Multi-modal Maneuver and Trajectory Prediction framework [MSKD23], a very recent work, comprises feature learning from vehicle data and lane markings and achieves this using a transformer-based encoder for context embedding and multimodal maneuver prediction. It then achieves trajectory forecasting using Gaussian distributions. It utilizes latent encoder representations and requires vehicle trajectories, road markings, and surrounding vehicle data as inputs.

However, most of the above-discussed methods had multiple modalities of information available to annotate the maneuver-changing events. Furthermore, even if maneuvers are identified, the data-driven methods should yield maneuver segments that are as long as the desired prediction horizon. Therefore, these constraints call for the use of clustering techniques along with deep representation learning. The learned reduced representations can be used for clustering, pattern finding, and similar use cases. [LL21] introduces a novel approach to representation learning through a deep autoencoder-based clustering framework. The objective function incorporates a clustering-weighted Mean Squared Error (MSE) loss, prioritizing the reconstruction of essential input feature values within the autoencoder. The proposed methodology is rigorously tested on the Human Activities and Postural Transitions (HAPT) Data Set, demonstrating a remarkable enhancement in K-means clustering performance ranging from 50% to 90%, with a distinct 30% improvement observed for the HAPT dataset.

More research is needed for the specific case of trajectory prediction for PTW [Con], let alone MTP. The authors of [JHAC23] introduce an Advanced Rider-cornering Assistance System based on Artificial Intelligence (AI) and NN techniques. They have developed new algorithms using the K-Nearest Neighbor (KNN) and ML techniques to estimate the maximum cornering velocity while taking in geometrical and inertial parameters and weather conditions. Another interesting study by the authors of [BVG18] focused on predicting how motorcycles overtake using Machine Learning methods. It used drones to gather detailed movement data and applied Decision trees and boosting algorithms for prediction. Besides PTW riders, there is research available for trajectory prediction of cyclists, another group of VRUs that share kinematics with PTW riders [HWP+21], [PKG21], [ZRK+22] yet again taking into account the surrounding context.

4 Proposed Methodology

The methodology proposed for this study initiates with the aim of assessing the viability of MTP for PTWs using vehicle dynamics data and distinct rider behavior signals as primary inputs. While the ultimate execution of MTP operates as an independent procedure, achieving an effective hybrid model hinges upon the outcomes and inferences derived from the preceding processes. The subsequent section delineates the sequential flow of this process

4.1 Process Flow

The initial step involved simplifying a complex time-series dataset featuring 16 intricate features using a suitable AE model. Following this, an in-depth exploration of the condensed data aimed to uncover distinct groups of clusters by leveraging an unsupervised clustering method. The goal here was to identify stable modes that persisted throughout the prediction time frame. These identified clusters were then utilized as labels for future trajectory modes extending across the entire prediction period. Subsequently, the plan involved training dedicated LSTM networks on these distinct modes or categorized data groups. Simultaneously, the development and training of a predictive classifier were initiated to recognize these specific modes within the data. The ultimate objective was to synthesize these individual components into a unified hybrid model that seamlessly integrates the dedicated LSTMs and predictive classifiers, forming the backbone for MTP. A brief overview of the proposed methodology is shown in Fig. 4.1.

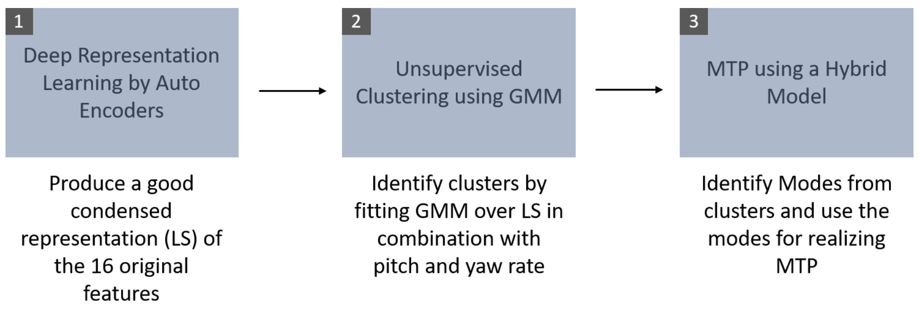

1
Deep Representation Learning by Auto Encoders
2
Unsupervised Clustering using GMM
3
MTP using a Hybrid Model
Produce a good condensed representation (LS) of the 16 original features
Identify clusters by fitting GMM over LS in combination with pitch and yaw rate
Identify Modes from clusters and use the modes for realizing MTP

Figure 4.1: Process Flow of the Proposed Methodology. The pitch and yaw rate used in step 2 are taken from the vehicle coordinate system

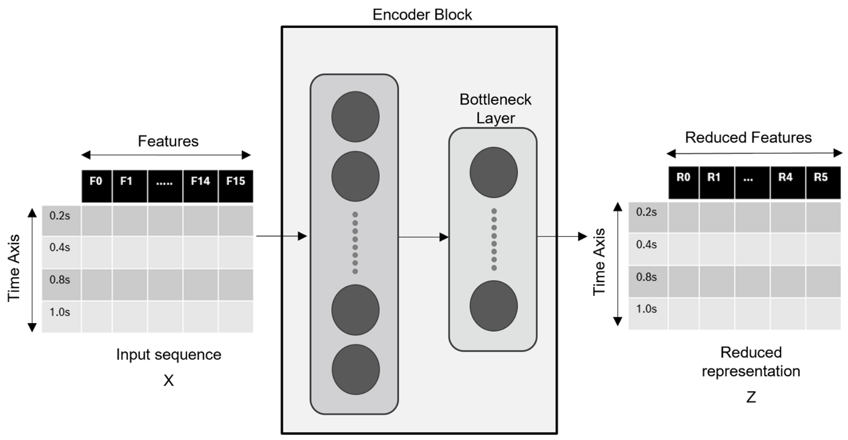


Figure 4.2: The rows of X and Z denotes the time axis. The time axis of each of these rows in X contains the 16 features associated with the time stamp. In this case, a time sequence starting from time stamp $t \in \{0.2s, 1.0s\}$ is considered.

4.2 Representation Learning of Time Series Data with AE

Since time series data have an inherent temporal data dependency across different time points, the input to an elementary AE network comprising linear layers should be passed as a time sequence rather than a single data point in time. Hence, time series data of n sequence lengths is the input to the AE model. Since the goal of identifying a reduced feature representation is to facilitate the identification of classes of longer sequences, the time series data is sampled at intervals of 200 milliseconds(ms). The choice of a 200ms sampling rate is rather empirical and is based on the notion that the shortest maneuver a rider can execute would be greater than half a second. A larger sampling rate aids in identifying patterns in data that span across longer time sequences while reducing the training time for deepAE networks. Therefore, each data point consists of 16 significant features sampled at 200ms. For instance, a 1-second input sequence translates to a 5 x 16 matrix, where the rows represent the time axis and columns denote the 16 features. Since we want to reconstruct the input sequence, the output sequence of the decoder takes the exact dimensions as the input sequence. On the other hand, the LS representation has the same number of rows but a different m number of columns rather than 16, where $m \in \{1, 10\}$. Fig. 4.2 illustrates the transitions an input tensor undergoes along the encoder of AE network, and Fig. 4.3 denotes the transformations the corresponding LS features undergo when passed to the decoder. Provided $n = 16$ and $m = 6$. The dimensions along the time axis are not reduced. This way, the model can access the temporal patterns across different time points while training. For this specific use case of identifying underlying patterns in our unlabeled time series data, reducing the dimensions along the time axis might lead to temporal information loss. The model can access the entire temporal history during training and make predictions by keeping the dimensions along the time axis. Hence, the dimensions along the time are retained throughout all the reconstruction processes in the following sections.

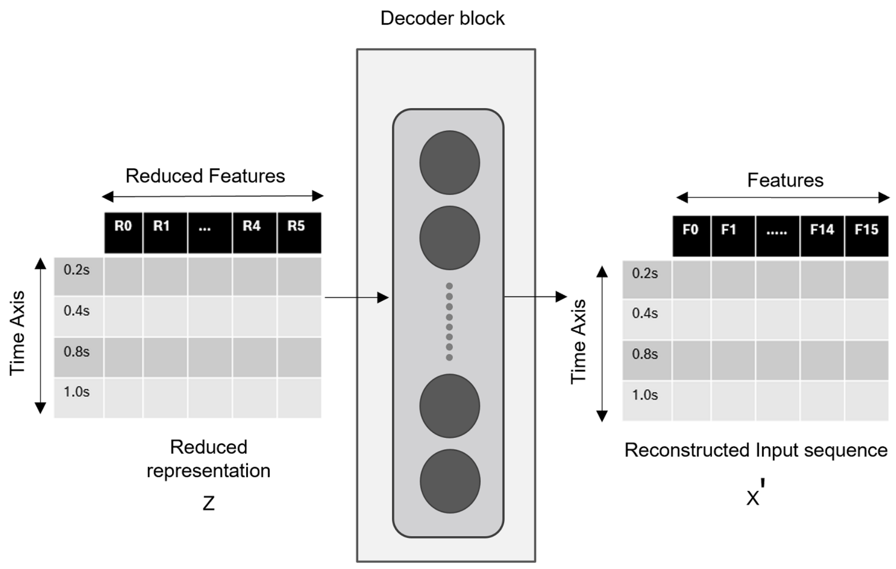


Figure 4.3: The columns of the reduced representation Z are the reduced features $R_j, j \in \{0, m\}$ (m= 5), learnt by the AE. Similar to X in 4.2, X' contains the 16 reconstructed features which are associated with the time stamp

4.3 Unsupervised Clustering with GMM

The choice of using GMM as the unsupervised clustering algorithm stems from the observation and inferences made on the LS produced from representation learning and are briefly discussed in sec. 5.5

4.3.1 GMM

GMM is a statistical model that can be represented as a combination of multiple Gaussian (normal) distributions, where each Gaussian distribution is a component of the mixture. This combination or mixture is created by assigning weights to each of these Gaussian components [HAK] and can be formulated by the Eq. 4.1.

$$\mathrm{P}(X \mid \Psi) = \sum_{i=1}^{M} w_i g\left(X \mid \mu_i, \Sigma_i\right) \tag{4.1}$$

X in Eq.4.1 represents the multivariate d dimensional LS vectors, and w_i represents the weights of the Gaussian mixtures associated with each of the M components. The function g represents the multivariate Gaussian function with d dimensions, where the densities are associated with a mean μ_i and covariance matrix Σ_i. The parameter Ψ represents the combination of mean vectors, covariance matrices, and mixture weights obtained from all individual component densities.

4.3.2 Labelling the Time Series Data

The primary objective of the clustering algorithm is to identify distinct cluster groups within the vehicle data by fitting the algorithm over the LS in combination with the pitch and yaw rate of the vehicle coordinate system. These two signals are chosen in particular because they have a high correlation with the global yaw rate, which in turn is very informative for encapsulating the road curvature [HBCY13]. Even though these signals are given to the AE, and their implicit information is available in the condensed representation. Due to several processes, such as normalization and the reconstruction of the time series data, their influence is reduced. Hence, the normalized version of these signals is once again added to the encoding produced by AE. Going forth, the term LS will refer to this combination of pitch rate and yaw rate from the vehicle coordinate signals along with the LS observed from the AE. There are inherent challenges associated with directly using the labels obtained from the GMM. Furthermore, a unique approach is taken for analyzing the results from clustering, and the details are discussed in chapter 6. The outcome of the clustering process is a label associated with each data point in time.

4.4 Proposed Hybrid Model for MTP

The hybrid model for MTP task employs a multifaceted approach to predict trajectories. Within this research context, the term 'multimodal' refers to the distinct characteristics of the observed vehicle dynamics data that remain consistent over the entire prediction horizon of 4 seconds. This is important because we aim to enforce a single consistent mode across the whole prediction horizon. The presence of multiple modes within the prediction horizon can lead to complications. Apart from model complexity, for this specific use case of MTP, labels were already generated using unsupervised learning. Hence, there is an inherent loss of accuracy stemming from minor occurrences of mislabelling. Designing more than one mode in the prediction horizon requires a cascading mechanism, where the input from one mode is the output of the other. Such an arrangement would cause a cumulative effect on the loss of accuracy of predictions from each mode in addition to supplementing model complexity.

The class labels identified from GMM are very short in time duration and do not span across the entire prediction horizon; hence a segregation logic is used to identify consistent modes across the prediction horizon by leveraging the modes defined by the segregation logic (see section 7.1), the dataset is segregated into distinct categories: DM1, DM2, DM3, and DM4, each representing specific observed conditions in the output time series sequence. The segregated datasets each have a dedicated modal LSTM model fine-tuned to the unique characteristics of each sub-dataset, allowing for nuanced predictions and learning from diverse scenarios. Thus, the segregation strategy introduces multimodality in prediction by enabling different modal LSTM models to specialize in handling specific trajectory patterns. This distinctive segregation forms the cornerstone of the model's multi-modal prediction strategy, augmenting its ability to predict diverse and complex trajectory outcomes. The collection of trained LSTMs is mapped to the modes they are trained on, and an LSTM predictive classifier is then used to give the future mode associated with the highest probability. The model architecture of the proposed hybrid model is shown in Fig. 4.4.

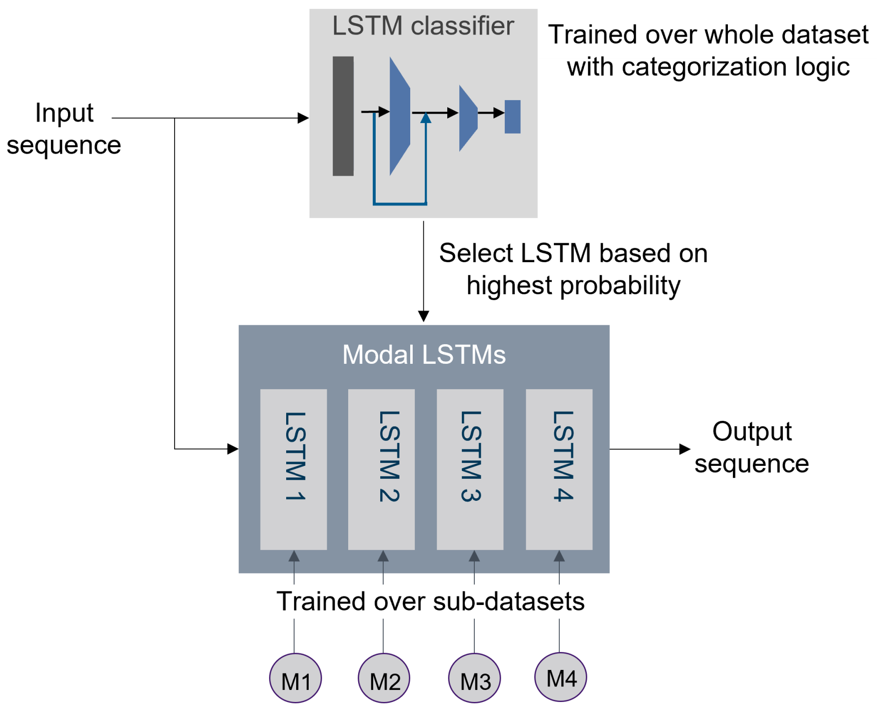

LSTM classifier
Trained over whole dataset with categorization logic
Input sequence
Select LSTM based on highest probability
Modal LSTMs
LSTM 1
LSTM 2
LSTM 3
LSTM 4
Output sequence
Trained over sub-datasets
M1
M2
M3
M4

Figure 4.4: The LSTM classifier and the collection of Modal LSTMs take in the same time series input. The roll angle predictions that are associated with a particular mode are given by the Modal LSTMs, and the classifier is in charge of selecting the modal LSTM that has the highest probability of occurrence

5 Realization of Representation Learning

Accurate data reconstruction is a crucial step in our pursuit of effectively discerning underlying patterns within our time series data. Hence, we acknowledge that determining the optimal architecture of AE for the dataset described in Section 2.1.1 remains an open question. Therefore, our research endeavors encompass a comprehensive exploration and empirical investigation into diverse AE models to unravel the most suitable approach. Therefore, we investigate a Deep AE architecture with Linear layers, AEs with LSTM, and BDLSTM.

5.1 Proposed AE Model Configurations

The choice of an appropriate reconstruction model is crucial for determining the quality and efficiency of the reconstruction process. We explore three variations of AEs and two proportions of bottleneck configurations for each of these AEs. Specifically, we consider a DAE, an LSTM AE (LSTMAE), and a BDLSTM AE (BDLSTMAE). The number of layers in each of these models plays a pivotal role in determining the dimensions of the LS. A model that achieves excellent reconstruction of the original data but results in a larger LS Dimension (LSD) may not be well-suited for clustering; it is important to strike a balance. Therefore, an ideal reconstruction model should yield a tolerable Mean Squared Error (MSE) and a smaller LSD. The definition of what constitutes a 'tolerable' MSE and an optimal LSD can only be confirmed during the clustering analysis and the final phase of multimodal trajectory prediction. Since there are many variables to consider and to ease the process of optimization, identifying the appropriate number of layers is isolated from fine-tuning the other hyperparameter; hence, two bottleneck configurations are considered. The bottleneck configuration with ten layers in total is referred to as the '**BN1**' configuration, and the one with six layers is termed as '**BN2**'; the layer proportions for each of these configurations for every variation of AE that are considered are detailed in Tab. 5.1 and Tab. 5.2. An overview of the different models that were explored is provided in Fig 5.1 and Fig. 5.2.

5.2 Experimental Setup for Time Series Reconstruction

The choice of sequence length in time series reconstruction is a critical factor that significantly impacts the model's ability to capture and recreate temporal patterns. A shorter sequence length can be advantageous when dealing with rapidly changing data. It allows the model to capture quick, localized patterns and is less prone to overfitting. Conversely, longer sequences provide a broader context for the model. They allow for capturing more extended temporal dependencies, making it possible to discern intricate patterns and relationships within the data. Since two-wheeler maneuvers are transient, the reconstruction model is trained over Input Sequence Length (IPL) of

	DAE			**LSTMAE**			**BDLSTMAE**		
Layer ID	**In**	**Hidden**	**Out**	**In**	**Hidden**	**Out**	**In**	**Hidden**	**Out**
1	I		H	I	H	H	I	H	2*H
2	H	-	H/2	H	-	H/2	2*H	-	H
3	H/2	-	H/4	H/2	-	H/4	H	-	H/2
4	H/4	-	H/8	H/4	-	H/8	H/2	-	H/4
5	H/8	-	L	H/8	-	L	H/4	-	L
6	L	-	H/8	L	-	H/8	L	-	H/4
7	H/8	-	H/4	H/8	-	H/4	H/4	-	H/2
8	H/4	-	H/2	H/4	-	H/2	H/2	-	H
9	H/2	-	H	H/2	-	H	H	-	2*H
10	H	-	I	H	I	I	2*H	I/2	I

Table 5.1: Layer proportions of AE models with BN1 configuration. I – input dimension, H – Intermediate layer size, L – Latent size

	DAE			**LSTMAE**			**BDLSTMAE**		
Layer ID	**In**	**Hidden**	**Out**	**In**	**Hidden**	**Out**	**In**	**Hidden**	**Out**
1	I	-	H	I	H	H	I	H/2	H
2	H	-	H/2	H	-	H/2	H	-	H/2
3	H/2	-	L	H/2	-	L	H/2	-	L
4	L	-	H/2	L	-	H/2	L	-	H/2
5	H/2	-	H	H/2	-	H	H/2	-	H
6	H	-	I	H	I	I	H	I/2	I

Table 5.2: Layer proportions of AE models with BN2 configuration. I – input dimension, H – Intermediate layer size, L – Latent size

1s, 1.6s, 2s, and 3s. With four values chosen for IPLs, a total of 24 model instances (three different types of AE models with two variations in each, followed by four different IPLs) are trained and evaluated.

Evaluation Metrics

MSE : The MSE is used as a loss function during training across different variants of AE. Moreover, MSE is sensitive to both the magnitude and direction of errors, which is crucial for assessing the quality of time series reconstructions, hence making it a suitable loss function. The same loss function is used over the validation dataset to evaluate the best values returned from each Optuna trial. The mathematical equation of MSE is shown in Eq. 5.1.

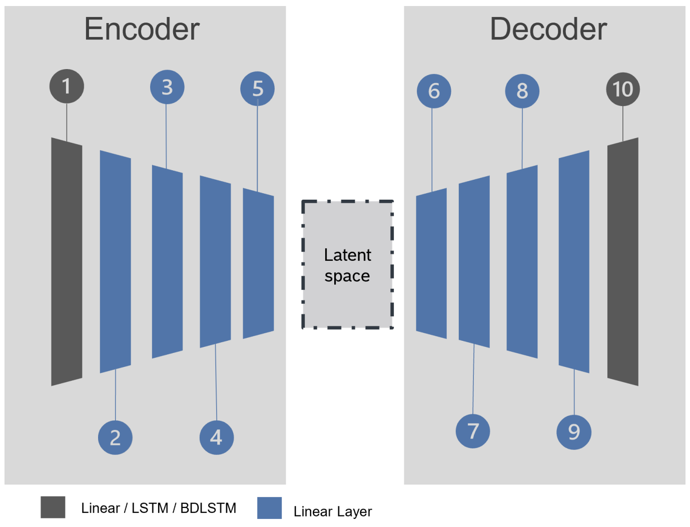


Figure 5.1: General template for AE models with BN1 configuration. The encoder and decoder contain a sequence of symmetrical layers that gradually decrease till the defined LSD, followed by an increase back to the original input size.

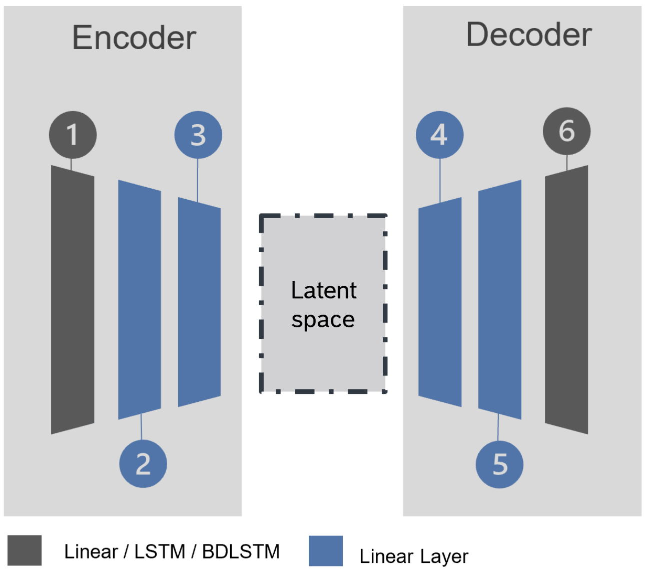


Figure 5.2: General template for AE models with BN2 configuration. The encoder and decoder follow a trend in proportion that is similar to BN1 model but with a reduced number of layers and a significantly smaller LSD

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{5.1}$$

R^2 value : The coefficient of determination, denoted as R^2, is a statistical measure that quantifies the proportion of the variance in the dependent variable (response variable y or ground truth) that is predictable from the independent variable (predictor variable x i.e., the vehicle dynamics features) [Uni23]. In the context of regression models, it provides insights into how well the model explains the variability in the response variable.

The coefficient of determination is calculated as the ratio of the regression Sum of squares (SSR) to the total sum of squares (SSTO). Mathematically, it is defined as Eq. 5.2[Uni23].

$$\begin{aligned} r^2 &= \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO} \\ SSR &= \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 \\ SSE &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ SSTO &= \sum_{i=1}^{n}(y_i - \bar{y})^2 \end{aligned} \tag{5.2}$$

The interpretation of R^2 is crucial in understanding the goodness of fit of a regression model. If $R^2 = 1$, it indicates a perfect fit, meaning that the predictor variable explains all the variability in the response variable. On the other hand, if $R^2 = 0$, the model does not describe any variability. Values between 0 and 1 represent the proportion of variability explained by the model, with higher values indicating a better-fit [Uni23]. In DL and regression models, R^2 can be a valuable metric for assessing the model's performance. DL models often involve complex relationships between input features and output predictions. The R^2 value helps us understand how much the model captures the variability in the target variable. A high R^2 suggests that the model effectively explains the variation in the output. In contrast, a low R^2 may indicate that the model needs to capture essential patterns or that the relationship between features and output needs to be better defined.

However, it is essential to note that while R^2 provides a measure of goodness of fit, it does not imply causation. In DL, where models can be highly complex and involve numerous parameters, other evaluation metrics, and interpretability techniques are often used in conjunction with R^2 to gain a comprehensive understanding of model performance. Hence, while MSE is used as the loss function for training our reconstruction models, each phase of the model during training, validation, and testing is evaluated by considering both the MSE and R^2 value. Hence, a smaller MSE and larger R^2 value are the basic trends that are considered for model evaluation.

Hyperparameters	**Exploration space**
Hidden State	$\{32, 64, 128, 256\}$
Learning Rate	$\{0.001 - 0.0001\}$
LSD	$\{3, 4, 5, 6, 78, 9, 10\}$
Dropout rate	$\{0.1 to 0.2\}$

Table 5.3: Parameters to optimize for BN1 configuration

Hyperparameters	**Exploration space**
Hidden State	$\{32, 64, 128\}$
Learning Rate	$\{0.001 - 0.005\}$
LSD	$\{3, 4, 5, 6, 7\}$
Dropout rate	$\{0.1 to 0.2\}$

Table 5.4: Parameters to optimize for BN2 configuration

5.2.1 Optimization

The optimization problem is defined for IPLs of 1s, 1.6s, 2s, and 3s sampled at 200ms with a standard batch size of 1024 for each of the AE models in both BN1 and BN2 configurations. The hyperparameters to fine-tuned are shown in Tab. 5.3 and Tab. 5.4

The Optuna framework was used for solving the optimization problem. Since BN1 models are proportionally larger and have a larger exploration space than the BN2 models, a set of 40 trials was carried out over the training dataset for BN1; on the other hand, BN2 models were subjected to 20 trials. The training loop for each above-mentioned optimization was set for a short range of 10 epochs.

Optimization Results : The tables 5.5,5.6, 5.7 and 5.8 show an overview of the chosen hyperparameters from the Optuna trials. The objective value signifies the MSE value calculated over the validation set during the optimization process. For training the different AE models for clustering, rather than selecting the least objective value as the only criteria, we also look at the suggested LS, and then select the trials that offer a balance between the two important factors.

Training and Validation

The DAE, LSTMAE, and BDLSTMAE models belonging to both BN1 and BN2 were subjected to 150 epochs of training. The parameters pertaining to the instance of the model that exhibits the most favorable MSE loss calculated during validation throughout the entirety of the training process were employed for subsequent testing. Both the training and validation phases shall be conducted using datasets that have undergone a normalization process. This normalization is imperative, as it

	BN1			BN2		
Hyper parameters	**DAE**	**LSTMAE**	**BDLSTMAE**	**DAE**	**LSTMAE**	**BDLSTMAE**
Hidden State	256	256	256	128	64	64
Learning Rate	0.0011	0.0022	0.0001	0.0012	0.0038	0.0041
LSD	9	9	6	7	7	7
Dropout rate	0.1	0.1113	0.1088	0.1268	0.1385	0.1016
Objective Value	0.16046	0.2871	0.3195	0.2724	0.2847	0.3218

Table 5.5: Optimization Results for IPL of 1.0 seconds

	BN1			BN2		
Hyper parameters	**DAE**	**LSTMAE**	**BDLSTMAE**	**DAE**	**LSTMAE**	**BDLSTMAE**
Hidden State	256	256	256	64	64	64
Learning Rate	0.0006	0.0003	0.0002	0.0018	0.0048	0.0021
LSD	10	5	8	5	6	7
Dropout rate	0.1122	0.207	0.1149	0.1507	0.1360	0.1414
Objective Value	0.0644	0.3103	0.3168	0.223	0.2852	0.3166

Table 5.6: Optimization Results for IPL of 1.6 seconds

	BN1			BN2		
Hyper parameters	**DAE**	**LSTMAE**	**BDLSTMAE**	**DAE**	**LSTMAE**	**BDLSTMAE**
Hidden State	256	256	256	64	64	64
Learning Rate	0.0002	0.0002	0.0003	0.0015	0.0024	0.0015
LSD	10	9	9	7	7	7
Dropout rate	0.1168	0.1089	0.1142	0.1003	0.1005	0.1635
Objective Value	0.0889	0.2935	0.3164	0.1357	0.2832	0.3199

Table 5.7: Optimization Results for IPL of 2.0 seconds

	BN1			BN2		
Hyper parameters	**DAE**	**LSTMAE**	**BDLSTMAE**	**DAE**	**LSTMAE**	**BDLSTMAE**
Hidden State	256	256	256	64	64	64
Learning Rate	0.0007	0.001	0.0001	0.0043	0.0034	0.0025
LSD	10	9	9	7	7	7
Dropout rate	0.1007	0.1005	0.1469	0.1004	0.1974	0.1849
Objective Value	0.0639	0.2923	0.3202	0.1438	0.2871	0.3193

Table 5.8: Optimization Results for IPL of 3.0 seconds

	BN1		BN2	
IPL	MSE	R^2	MSE	R^2
1.0s	**0.0451**	**0.9548**	**0.1163**	**0.8837**
1.6s	0.0476	0.9524	0.1981	0.8019
2.0s	0.0480	0.952	0.1329	0.8674
3.0s	0.0457	0.9542	0.13	0.87

Table 5.9: Evaluation of DAE models for reconstruction. The values in bold have the lowest MSE and highest R^2 value and correspond to the model with the best reconstruction results

	BN1		BN2	
IPL	MSE	R^2	MSE	R^2
1.0s	0.3156	0.6849	0.3262	0.6742
1.6s	0.3395	0.6607	0.3298	0.6705
2.0s	**0.3025**	**0.6970**	0.3164	0.6846
3.0s	0.3031	0.6972	**0.3148**	**0.6853**

Table 5.10: Evaluation of LSTMAE Models for reconstruction. The values in bold have the lowest MSE and highest R^2 value and correspond to the model with the best reconstruction results

aligns with the primary objective of the reconstruction model: the production of an LS conducive to clustering analysis. The specific equation applied for the normalization procedure is shown in Eq. 5.3.

$$Z = \frac{x - \mu}{\sigma} \tag{5.3}$$

5.3 Reconstruction Results and Inference

The trained model instances were tested on the test set and the results are shown in Tables 5.9, 5.10 and 5.11

Starting with the evaluation metrics, DAE demonstrated optimal reconstruction performance with an IPL of 1.0 seconds, as indicated by the lowest MSE and highest R2 values for both BN1 and BN2 configurations. However, its performance degraded with longer input sequences, particularly in the BN2 configuration. In contrast, both LSTMAE and BDLSTMAE exhibited robust performance with increasing IPLs, with BN2 consistently outperforming BN1 across all evaluated lengths. This trend suggests that a bottleneck configuration with fewer symmetric layers may more effectively capture patterns in longer-time sequences.

	BN1		BN2	
IPL	**MSE**	R^2	**MSE**	R^2
1.0s	0.3049	0.6953	0.3232	0.6771
1.6s	0.3018	0.7038	0.3175	0.6827
2.0s	**0.296**	**0.952**	**0.312**	**0.6867**
3.0s	0.3025	0.6977	0.313	0.6871

Table 5.11: Evaluation of BDLSTMAE Models for reconstruction. The values in bold have the lowest MSE and hig R^2 value and correspond to the model with the best reconstruction results

	BN1			BN2		
IPL(s)	**DAE**	**LSTMAE**	**BDLSTMAE**	**DAE**	**LSTMAE**	**BDLSTMAE**
1.0	92185	1465993	3618441	21654	101639	68359
1.6	92314	1465477	3618184	6677	101510	68359
2.0	92314	1465993	3618441	6807	101639	68359
3.0	92314	1465993	3617927	6807	101639	68359

Table 5.12: Total Number of Trainable Parameters

The examination of trainable parameters (see Tab. 5.12) provides additional insights. Surprisingly, BN2 configurations consistently performed almost equally to their BN1 counterparts despite having a significantly lower number of trainable parameters. This observation suggests that the BN2 configurations strike a balance between model complexity and performance. Specifically, DAE model with BN2 demonstrated competitive reconstruction results with a substantially reduced number of trainable parameters compared to BN1, emphasizing the efficiency of the simpler architecture. The LSTMAE and BDLSTMAE models followed a similar trend, with BN2 consistently performing on par with BN1, showcasing the effectiveness of a less complex configuration in capturing underlying temporal vehicle dynamics.

In summary, our experiments highlight the nuanced relationship between IPL, model architecture, bottleneck configuration, and the number of trainable parameters in the context of time series reconstruction. While DAE excels in short-term reconstruction, LSTMAE and BDLSTMAE demonstrate enhanced performance with longer input sequences, and the BN2 configurations consistently perform on par with or even outperform their BN1 counterparts, showcasing a remarkable balance between simplicity and efficacy in time series reconstruction.

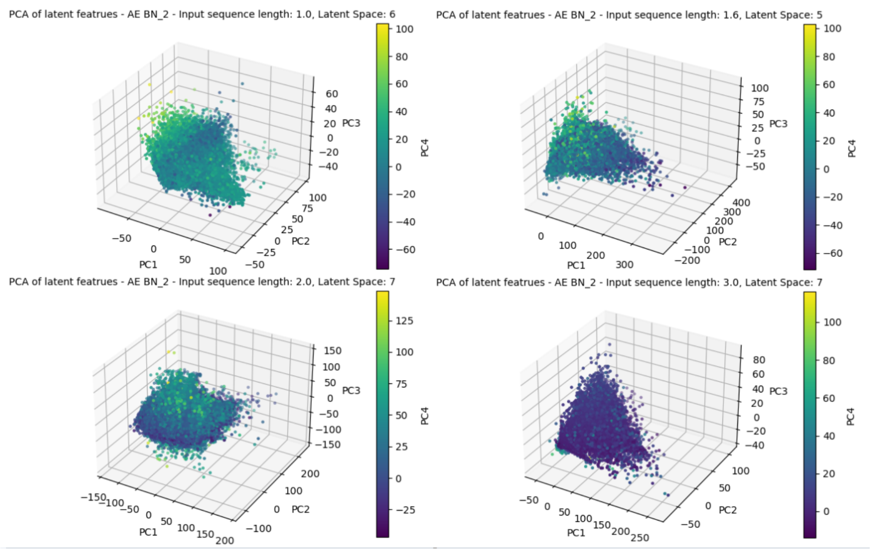


Figure 5.3: Four component PCA visualization of LS from DAE BN1 models. The PCs do not have any distinct group of clusters that are visible in three dimensions

5.4 Observations on the LS produced by the AE Models

Understanding the characteristics of the LS produced by the different AE models is pivotal since they serve as the repository for condensed representations of our input data. The LS features generated by different variations of AEs discussed in the previous sections typically range from 5 to 10 dimensions. Despite reducing the feature space from the original 16 to a range within $\{5, 10\}$, visualizing these dimensions remains challenging. Addressing this visualization obstacle necessitates effective dimensionality reduction while preserving vital information. Principal Component Analysis) (PCA) emerges as a foundational and widely utilized technique precisely for this purpose. PCA simplifies datasets by extracting new variables, known as principal component (PC), to maximize data variability. Relying on eigenvalue/eigenvector solutions, PCA generates PCs adaptable across various datasets, offering descriptive utility independent of specific data distributions and suitable for diverse numerical data types.

The LS features from the AEs are reduced to four PCs to facilitate a more accessible and insightful visualization, and this is done by using the PCA implementations available in the scikit-learn library. The PCs are visualized as a 3d scatter plot, where the 1st three PCA components delineate the x, y, and z axes, capturing the essential spatial relationships within the LS. We used a color-coded representation to convey the information encapsulated in the fourth PC. The visual PCA representations of the LS produced across the BN1 and BN2 configurations for DAE models are shown in Fig. 5.3 and Fig. 5.4.

The reduced visual representation of LS across all models reveals a complex, overlapping, and sometimes indistinguishable structure unique to each model variation. This distinctive feature underscores the challenge of straightforward, distinct clustering. Furthermore, the LS's inherent complexity

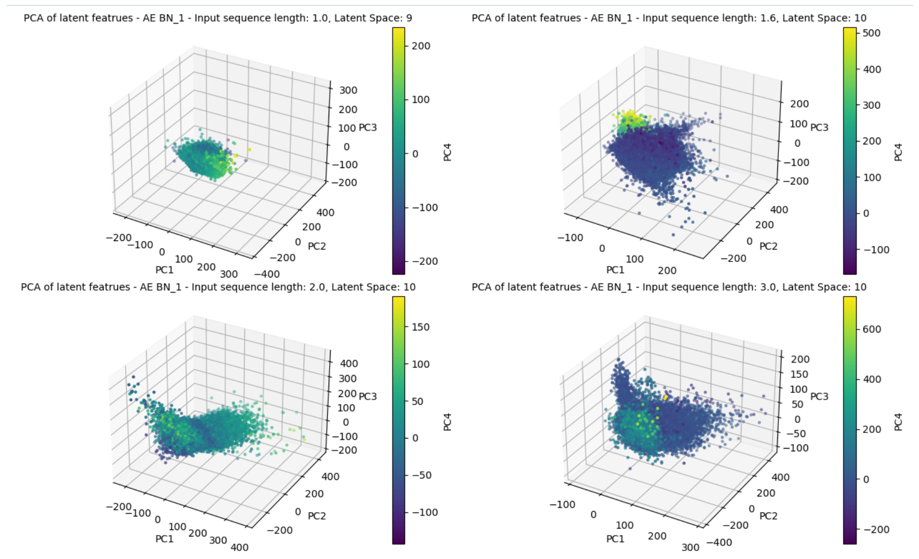


Figure 5.4: Four component PCA visualization of LS from DAE BN2 models. The PCs do not have any distinct group of clusters that are visible in three dimensions

complicates the application of traditional clustering methods, as observed data points lack clear, well-defined boundaries, diminishing the effectiveness of distance metric-based methods such as K-means.

While PCA offers an effective means to visualize the multidimensional LS and identify the spatial properties of the PCs, its sole utilization might be limiting in comprehensively understanding the data's intrinsic structure and distributional characteristics. PCA primarily focuses on capturing the variance and maximizing data dispersion along principal axes. However, further exploration beyond PCA becomes imperative to uncover a deeper understanding of the data's underlying complexity and potential multimodal nature. Consequently, identifying the probability distribution of each feature of the reduced LS allows for a more nuanced examination of individual feature distributions.

Probability density function (PDF) accurately illustrates probability mass distribution across a random variable, commonly explored through histogram representation given sample data points. However, histograms, reliant on chosen intervals (bins) and prone to subjectivity, exhibit limitations due to binning issues. These constraints introduce discontinuity and flatten data within bins, impacting their fidelity in representing the actual data. Conversely,Kernel Density Estimation (KDE) serves as an alternative technique circumventing these limitations. By constructing a smoother empirical PDF based on individual data points, KDE mitigates binning issues, offering a more accurate representation of the underlying data distribution [Węg18].

The KDE implementations are available in the sci-kit-learn library are used to visualize the distribution of each feature in the LS. The said KDE is generated in a univariate fashion, where each PDF generated for a feature is independent of the other. Our choice of using univariate KDE arises from practical constraints. While there are methods to perform bivariate KDE from sci-kit-learn, our LS ranges from $d \in \{5, 10\}$, which implies that we have to analyze around 2^d KDE for a single AE configuration and we have 24 such AE variations, which would amount to 24 x 2^d KDE plots. The challenge is further complicated because our LS features are unlabeled, and we perform the clustering in an unsupervised fashion. Therefore, to reduce the computation scale and complexity, a univariate KDE for each of the LS variables is visualized, and the KDE plots for DAE BN2 configuration with IPL of 1.0s and 1.6s are shown in Fig. 5.5 and Fig. 5.6

Interpreting KDE plots is crucial in understanding the distributional characteristics within an LS. In the context of LS analysis, a KDE plot showcases the probability density function of individual features. When a KDE plot displays two to three significant peaks, it typically signifies the potential existence of multiple modes or distinct subgroups within that feature's distribution. Each peak represents a mode, indicating high-density regions where data points cluster. Having multiple peaks suggests the presence of distinct subgroups or clusters. Conversely, a KDE plot exhibiting a single prominent peak doesn't necessarily negate the presence of subgroups; it could still contain crucial information essential for accurate reconstruction or represent a unique characteristic within the LS. Furthermore, it's crucial to note that the observed peaks are derived solely from univariate KDE plots, representing distributions within individual LS features. The absence or presence of peaks in these univariate KDEs doesn't conclusively define the absence or presence of subgroups on the multivariate level. The multivariate KDE, which integrates information from all features simultaneously, might reveal nuanced relationships and subgroups that are not evident in individual univariate visualizations. Therefore, while insightful, the univariate KDEs alone cannot fully elucidate the complexity of the LS, necessitating a cautious interpretation.

Exploration of LSs derived from different AE configurations revealed diverse characteristics. AE BN2 with an IPL of 1.6 seconds demonstrated the presence of three significant peaks in all but one of its LS features, suggesting the likelihood of multiple subgroups or clusters within each feature's distribution. Despite this, models showing single-peaked KDE plots, such as AE BN2 with IPL of 1.0s and certain LSTM-based models, offer valuable information crucial for accurate reconstruction. However, considering the complexity and potential multimodal nature of the data, exploring all 24 models becomes imperative. Each model configuration showcases unique aspects of the LS, offering insights essential for clustering and multimodal prediction tasks. This comprehensive exploration ensures a robust understanding of the LS's diverse characteristics, contributing to better-informed decisions in subsequent clustering analyses and multimodal predictions.

5.5 Choosing an Appropriate Clustering Algorithm

The application of clustering techniques to data having more than three dimensions poses a unique set of challenges, particularly in scenarios where the dimensionality of the LS cannot be further reduced. In such instances, it is imperative to select an appropriate clustering algorithm that is well-suited to the complex and multidimensional nature of the data. One fundamental consideration in choosing a clustering algorithm for multi-dimensional data is the capacity of the algorithm to transcend traditional geometric limitations. In the context of clustering based on distance, the choice of a suitable distance metric in high dimensions is often not straightforward, and the widely used distance metrics like Euclidean distance and its relatives may not perform well for such data points, Moreover, L_k norm is more suitable with k values less than or equal to 2 and is not accurate for $k >= 3$, hence ruling out the possibility of considering other metrics, since, the meaningfulness

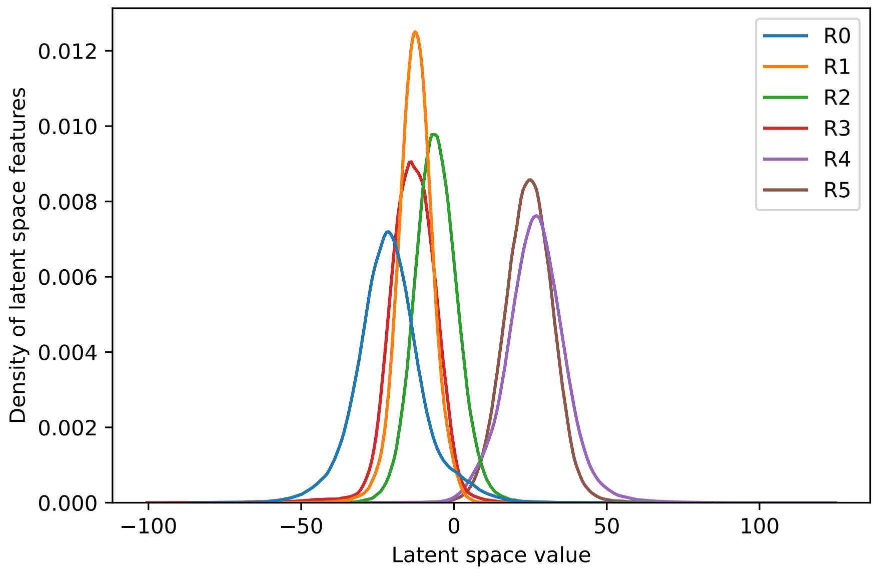


Figure 5.5: KDE of the LS features produced by BN2 DAE model with IPL 1.0s. R represents the features in the LSD

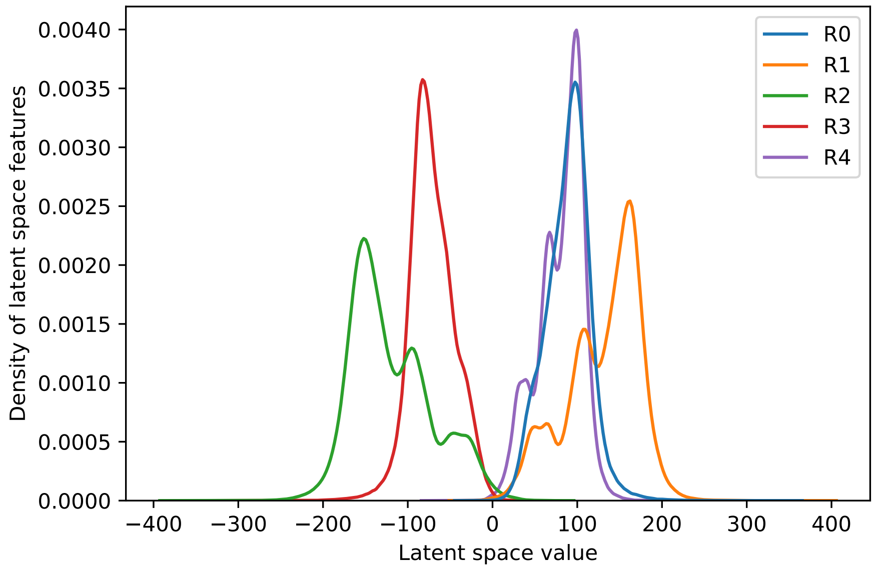


Figure 5.6: KDE of the LS features produced by BN2 DAE model with IPL 1.6s. R represents the features in the LSD

of distance measures appears to degrade as dimensionality increases, especially with higher values of k[AHK01]. To address this issue, fractional distance metrics with k values less than one are proposed. But these metrics are more useful only when we know which dimension of the LS contributes the most for clustering, and since we are considering the vectors in the LS, which are the encoding from an AE, identifying which dimension is crucial is a much more difficult task since we don't really know which dimension of the LS represents what. Hence, it's better to consider clustering algorithms not based on distance.

GMM, however, does not solely rely on geometric assumptions. Instead, it leverages a probabilistic framework to model the distribution of data points in the LS. This characteristic makes GMM particularly well-suited for capturing complex structures in data that may not conform to simple geometric patterns, which is precisely the behavior observed in our LS: complex overlapping cluster with no clear boundary. Furthermore, the decision to avoid density-based clustering methods is rooted in the need for consistent clustering results across different rider files. The data in these files exhibits vast variability, making it challenging to determine a uniform density threshold for density-based clustering. As the data density can vary significantly from one file to another, density-based methods might lead to inconsistent clusters that do not align with the inherent nature of the data. To ensure that the same nature of clusters is identified across all files, GMM's distribution-based approach, which is independent of data density, presents a compelling alternative.

Another compelling feature of GMM is its flexibility in accommodating various data distributions. By assuming that data points are generated from a mixture of Gaussian distributions, GMM can identify clusters in data with varying shapes and densities while avoiding the pitfalls associated with density-based methods. This adaptability is especially beneficial when working with data points in the LS of an AE, as this space can exhibit intricate and heterogeneous patterns that may not adhere to a single, predefined distribution.

In conclusion, the GMM emerges as a better choice for the clustering of multi-dimensional data originating from the LS. Its capacity to transcend geometric constraints, its adaptability to diverse data distributions, and its inherent ability to estimate the number of clusters makes it a valuable tool for uncovering hidden structures in complex datasets.

6 Clustering with GMM

The successful implementation of GMMs necessitates a structured approach that involves several key considerations. Firstly, determining the appropriate number of Gaussian components, similar to bell-shaped curves, is a fundamental decision in the GMM modeling process. Subsequently, the fitting of the GMM model to the data is paramount. The primary objective of this step is to fine-tune the parameters of the Gaussian components to best align with the observed data, thereby enabling the model to encapsulate the inherent patterns and structures within the dataset effectively [Rey09].

6.1 Experimental Setup

In order to estimate accurate clusters and generate consistent class labels for each cluster, the chosen GMM algorithm has to be fitted over the entire dataset (including validation and test sets). In an unsupervised approach, the scikit-learn implementation of GMM algorithm is recommended only when there are less than 10,000 samples[PVG+11], but the total number of samples across the entire dataset amounts to 1333679. The mclust package in R doesn't have an explicit sample size limit as in the sci-kit-learn GMM implementation. Hence the clustering is performed using the mclust package in R studio.

Fixed Number of Gaussian Components

When deciding the number of components for GMM, it is common practice to employ model selection criteria like the Bayesian Information Criterion to determine the optimal number of clusters. However, our specific application, which involves multimodal trajectory prediction using separate LSTM models for different data categories, necessitates a different approach. We have chosen to deliberately use a fixed number of three clusters. This decision is driven by the inherent nature of our unlabeled dataset and the need to establish clear, predefined clusters that directly correspond to the categories our LSTM models address. While traditional methods may suggest alternative cluster counts, our choice of three clusters is carefully tailored to align with the unique demands of our MTP task. This approach ensures that our clustering structure serves the primary goal of category separation, enabling practical subsequent model training and interpretation.

Data Preparation

The output derived from the encoder of the trained AE models comprises the condensed representation for every data point within the entire input sequence of varying lengths: 5 for 1.0s, 8 for 1.6s, 10 for 2.0s, and 15 for 3.0s. The pre-processing stage for the Gaussian Mixture Model (GMM) entails the selection of the current timestamp from each input sequence, ensuring a direct correspondence between data points in the original feature space and the reduced LS. Specifically, the current data point corresponds to the final time sample in the input time series sequence, and similarly, its reduced representation is derived from the last and final time sample from the output sequence of the AE's encoder. Given that the dimensions have been reduced from the original 16 to a range of $d \in \{5, 10\}$, the outputs obtained from the AE's encoder do not undergo further pre-processing and are directly utilized for fitting using the GMM algorithm.

6.2 Challenges in Evaluation of GMM Output

Evaluating the quality of clusters generated by GMM in the context of vehicle dynamics data presents a set of distinct challenges. The foremost challenge stems from the inherent complexity of the data, where clusters are often irregular in shape and exhibit significant overlap. The multi-dimensionality of the LS vectors, comprised of n variables, further complicates the task, making it impractical to employ traditional cluster evaluation methods that rely on labeled data or the assumption of well-separated, convex clusters. In response to these formidable challenges, Cluster profiling and visual interpretation techniques are employed. The proposed evaluation strategy encompasses a 3D scatter plot visualization that considers statistical measures (mean, skewness, and standard deviation) of the LS variable while cross-validating the clustering results with their GPS coordinates. This method offers a data-driven, intuitive approach to overcoming the intricacies of cluster evaluation, particularly in the context of vehicle dynamics data analysis, where traditional evaluation practices fall short in addressing the complexities of overlapping and irregularly shaped clusters.

6.3 Addressing the Label Switching Problem

When dealing with GMMs, a crucial challenge arises, known as the "label switching problem."The label-switching problem occurs in unsupervised learning tasks, particularly in mixture models like GMMs. It arises due to the inherent identifiability issues within these models, leading to multiple equivalent solutions that exhibit the same likelihood but differ in the assignment of cluster labels. This means that the clustering solution remains the same in terms of probability density estimation, but the labels assigned to each cluster interchange or switch among different runs of the algorithm [Mur12]. For instance, when fitting a GMM to a dataset, the identified clusters may be represented differently in various model runs, resulting in an inconsistency of labels across these runs. This phenomenon poses a challenge in interpreting and comparing the results obtained from different GMM fittings, especially when seeking consistency or reproducibility in clustering assignments.

Several well-established techniques are available to address label switching in GMMs, including Identifiability Constraints, Initialization Techniques, Posterior-Based Re-labeling, Permutation-Invariant Metrics, and Bayesian Approaches [Ste00]. However, a simpler approach was chosen in this study for specific reasons. The complexity and potential intricacies associated with these established techniques necessitate meticulous tuning and may not always guarantee a straightforward resolution to the label-switching problem within the context of identifying patterns in the vehicle dynamics dataset. Instead, a more intuitive and visually interpretable method was favored, involving the profiling of basic statistical properties within the identified groups. This approach entails the examination and analysis of statistical descriptors, such as mean, standard deviation, and skewness, within each identified cluster. These statistical properties were mapped onto a 3D scatter plot, facilitating differentiation and interpretation of the clusters based on their distinctive statistical characteristics. Moreover, the proposed simpler approach allows for a more direct and transparent interpretation of the clustering results. It emphasizes intuitive insights derived from basic statistical properties, enabling a clearer comparison across different models or runs. By opting for visual interpretation and the profiling of statistical properties, the aim was to achieve a practical and comprehensible solution to the label-switching problem. This approach aligns with the specific needs and objectives of this analysis, prioritizing clarity and ease of interpretation over complex technical methodologies.

6.4 Statistical Profiling and Relabeling of GMM Clusters

Addressing label switching ensures robustness and reliability in clustering analyses, particularly when comparing results across various model configurations or experiments. In the pursuit of identifying optimal clustering in variably encoded LSs produced by the 24 distinct AE models, each exhibiting different LSDs, statistical profiling methodology is chosen to achieve uniformity in labeling and consistent cluster interpretation. The objective entails fitting GMM independently across all LSs derived from different AE configurations, restricting GMM components to three to delineate three distinctive classes.

The proposed method of statistical profiling is adopted post-GMM clustering and involves individually calculating the mean, Standard Deviation (SD)), and skewness for each LS feature individually within every identified cluster (class 1, class 2, and class 3). This analysis generates $3xn$ data points, where n signifies the LSD. The next step involves mapping these computed statistical descriptors onto a 3D scatter plot, where class differentiation manifests spatially: SD values along the x-axis, mean along the y-axis, and skewness along the z-axis. The 3D scatter plots displaying the simple statistical features of the LS features within a cluster group before relabeling are shown in Fig. 6.1.

In conducting this process across 24 models, a notable observation emerged upon close examination. It was consistently observed that within each set of scatter plots, there existed a cluster group where more than half of the LS features exhibited a notably higher SD, another cluster group where more than half of the features displayed the smallest SD and a third cluster group where more than half of the features lay in the mid-range of SD. This recurrent pattern across the 24 scatter plots led to identifying distinguishable clusters based on their SD. Given this consistent pattern observed across different models, it was discerned that SD is a reliable indicator for grouping similar clusters. A prevailing trend emerged where specific cluster groups consistently exhibited distinct SD profiles. As such, these recurrent observations substantiated the decision to employ SD as a criterion for identifying similar groups of clusters. The application of SD to differentiate and establish common labeling schemes across clusters was guided by this empirical finding, ensuring consistency and coherence in the labeling process.

The proposed algorithm is depicted in Algo. 6.1 aims to establish the relative positions of clusters within a dataset by leveraging the SD values calculated for each feature within distinct cluster groups. This algorithm delineates three cluster labels, namely Class 1, Class 2, and Class 3, based on the comparative analysis of SD arrays computed for the clusters. Algo. 6.1 begins by computing the SD arrays for each cluster, representing the variability of features within the respective clusters. The algorithm then systematically compares the SD values across LS features, assigning positions to clusters based on their SD magnitudes. Initially, the algorithm constructs two vote arrays, HighestSDVotes and MidSDVotes, to determine the classes with the highest and second-highest SD column-wise across LS features. Subsequently, the algorithm identifies the winning class, the second-highest class, and the remaining class with the least SD. In cases of tie-breaking for the highest or second-highest SD, additional procedures are executed via the ResolveTie function. The ResolveTie function embedded within the algorithm serves as a crucial mechanism for resolving tie scenarios encountered during comparisons of SD values among classes. For the highest SD tie situation, it conducts a comprehensive comparison of the maximum SD values within the tied classes across all LS features. Subsequently, it assigns the class with the higher maximum SD as the prevailing winning class and designates the class with the lower maximum SD as the second class.

Similarly, in cases of a tie for the second-highest SD position, the function follows an analogous methodology to determine the appropriate positioning by evaluating the maximum SD values across LS features and allocating the classes accordingly. This process ensures the accurate delineation of cluster positions based on SD comparisons, maintaining clarity in hierarchical arrangements amid tie situations. If a tie persists across all three positions, the algorithm resorts to nominating features with the highest SD values within each class and decides the cluster positions based on these nominated features. Finally, the algorithm returns the positioning based on the SD comparison, indicating the relative order or ranking of clusters according to their SD characteristics.

Leveraging SD as a consistent criterion for label assignment, the grouping is standardized: clusters associated with high SD are designated as 'class 2,' medium SD as 'class 1,' and low SD as 'class 3.' This meticulous approach ensures uniform labeling across diverse LSs derived from various AE models, facilitating consistent interpretation and comparative analysis.

6.5 Results: Visualizing Cluster Labels with GPS Coordinates

Within the framework of this research, deriving meaningful interpretations from clustered data points related to vehicle dynamics is essential. A visualization strategy has been devised to facilitate a comprehensive understanding of these clusters. This strategy involves mapping their spatial distribution across the geographical region where the rides were recorded, utilizing the

Algorithm 6.1 Determining Cluster Positions Based on Standard Deviation

```
function SDPositioning(Class1SDArray, Class2SDArray, Class3SDArray)
    HighestSDVotes, MidSDVotes ← empty_arrays
    for col in range(LSD) do
        Compare and assign highest SD class label to HighestSDVotes[col]
        Compare and assign second highest SD class label to MidSDVotes[col]
    end for
    if WinningClass.tie() with SecondHighestClass for highest SD then
        ResolveTie(WinningClass, SecondHighestClass)
    end if
    if SecondHighestClass.tie() with RemainingClass for second highest SD then
        ResolveTie(SecondHighestClass, RemainingClass)
    end if
    if Tie for all three positions then
        NominateFeaturesWithHighestSD()
        DecidePositionsBasedOnFeatures()
    end if
    Return: Positioning based on SD comparison
end function
function ResolveTie(Class1, Class2)
    Compare highest SD values across all LSD columns within Class1 and Class2
    Assign higher SD class to WinningClass, lower SD class to SecondClass
end function
```

inherent GPS coordinates incorporated into the GMM dataset derived from the measurement file. Subsequently, updating the dataset with appropriate cluster labels enables the visualization of the clusters' spatial arrangement. This visualization initiative aims to offer valuable insights into the clustered data points within the context of vehicle dynamics, thereby enhancing our understanding of their geographical implications. Given the extensive nature of the total ride, which spans 65 hours of time series data across 77 files, the GPS visualizations given in this section focus solely on how the LS encodings from DAE models trained for an IPL of 1.0s and 1.6s influence clustering across specific ride scenarios: (a) Highways, (b) Urban roads, (c) Round about and (d) rural roads. These specific IPLs are chosen simply because they demonstrate different LSDs 5, 6, 9, 10. The said visualizations are shown in Figures: 6.3, 6.4,6.5 and 6.6.

Examining the clustered data from our GMM dataset reveals some distinct trends. Class 3 data points often appear during mostly straight-ride sections, although there are occasional exceptions, and such occurrences of class 3 sections were more frequent in rides recorded within the cities and on the highways. This implies that the occurrence of class 3 could be influenced by traffic regulations (for instance, speed limit, restricted turns)(refer Fig. 6.6). Class 2 instances, on the other hand were prominent in rural roads (see Fig. 6.5) and were more often found at sharp, but they sometimes occur along straight paths and a similar trend was observed for class 1. Further exploration is needed to understand why some LS features corresponding to straight ride trajectories show higher variability, causing the GMM to assign these instances to class 2. Notably, data in class 1 typically corresponds to smoother turns or wider curves along the ride's path. It's important

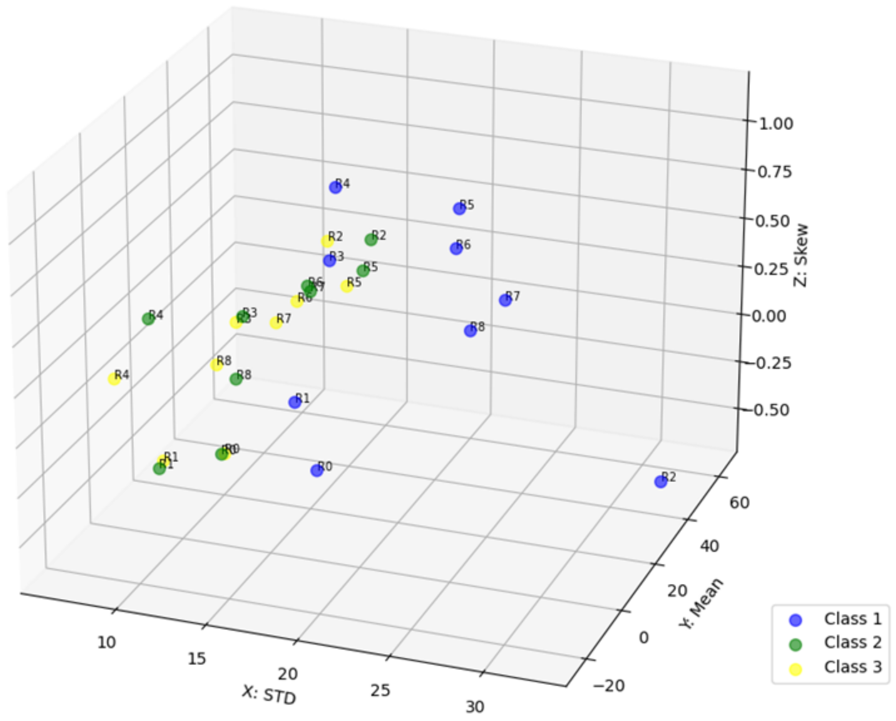
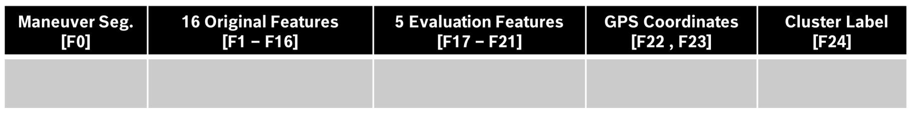


Figure 6.1: Statistical Profile of LS Features from AE BN1 Model for IPL 1.6 s



Maneuver Seg. [F0]	16 Original Features [F1 – F16]	5 Evaluation Features [F17 – F21]	GPS Coordinates [F22 , F23]	Cluster Label [F24]

Figure 6.2: GMM Dataset

to note that while looking at these traits, not all AE LS features consistently give the same labels for ride paths. Instead, there's a noticeable trend rather than an exact match(refer Figures: 6.3, 6.4, 6.5 and 6.6).

These observations hint at potential correlations between ride characteristics and how the GMM classifies different patterns in vehicle dynamics. In the context of the AE BN2 model trained for an input length of 1.6 seconds, distinct categorizations were observed among the assigned classes. Class 1 was attributed to ride segments positioned closer to the higher extremities of road curvatures, rather than its other IPL counterparts, where class 1 identified trajectories with moderate road curvature. Instances designated as class 2 corresponded to fine-tuned ride trajectories characterized by the highest road curvatures or markedly sharp transitions (referred to as abrupt changes in dynamics).

Moreover, though class 2 identified trajectories that corresponded to sharp turns and rapid changes in dynamics, this range was smaller than its other IPL class 2 counterparts. Revisiting the SD profiles for these classes revealed closely aligned standard deviation SD values within the LS features of class 1 and class 2, evident in the scatter plot visualization shown in Fig. 6.7. This observed trend persists across other models, although the discussion here focuses solely on this

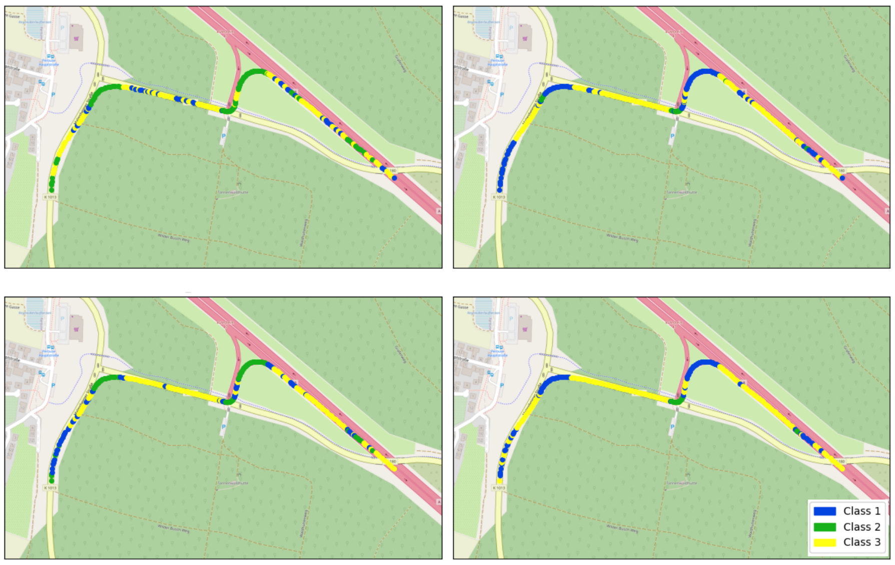


Figure 6.3: GPS visualization of clusters identified using DAE models for a Highway scenario. (a) Top left: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 9, (b) Top right: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 10, (c) Bottom Right: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 6, (d) Bottom Left: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 5.

specific case for brevity. Consequently, the distinctiveness of class 3, class 2, and class 1 depend significantly on the SD differences among them, providing essential insights into the clustering tendencies observed across various models.

6.6 Inference: Analyzing Vehicle Dynamics within Identified Clusters

To further comprehend the cluster labels derived from the GPS visualization, an analysis of the original vehicle dynamic features within these identified clusters was conducted. While the GPS visualization provided valuable insights, it only encompassed some aspects. Hence, to delve deeper into the correlation between vehicle dynamics and the assigned cluster labels, the focus was placed on seven specific original features: 'Roll Angle', 'Lateral acceleration (from vehicle coordinate system),' 'Steer Angle,' 'Steer Rate,' 'Roll Rate,' and 'Rider Lean Angle.'

These features were chosen as they potentially influence the classifications, primarily when trajectories seemingly straight were assigned to clusters based on high variability across LS features. In addition to impacting classifications, these lateral vehicle dynamics also play a crucial part in the original trajectory prediction problem. It is important to note that out of the 16 original features, only these six were analyzed across all 24 models to understand their relation to the identified clusters better. The analysis of vehicle dynamics within identified clusters involved employing a fundamental yet insightful visualization tool: the box plot. This approach was pivotal

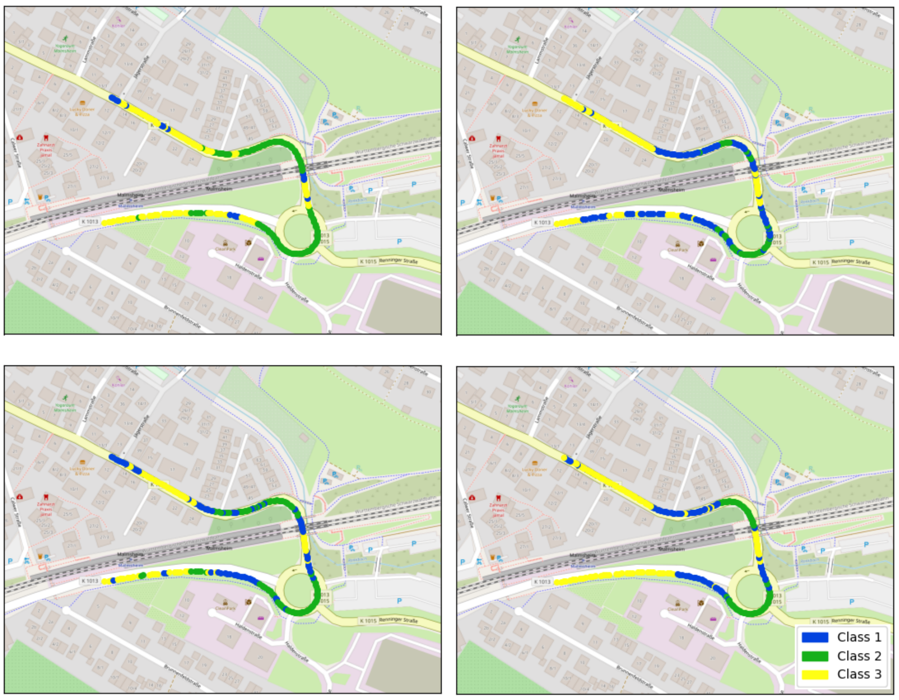


Figure 6.4: GPS visualization of clusters identified using DAE models for a round-about scenario. (a) Top left: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 9, (b) Top right: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 10, (c) Bottom Right: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 6, (d) Bottom Left: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 5.

in comprehending the behavior and distribution of selected vehicle dynamic features across the entirety of 24 distinct models. By utilizing this straightforward graphical representation, a detailed examination of the chosen features' characteristics within the context of various models was achieved. This method provided a clear depiction of the spread, central tendency, and potential outliers within each feature, enabling a comprehensive comparative analysis across the diverse model configurations.

After conducting a comprehensive analysis using box plots across the 24 distinct models, it's evident that some lateral vehicle dynamic features exhibit consistent trends within the identified clusters. Steer angle, steer rate, lateral acceleration, and rider lean angle, although showing discernible patterns similar to roll angle and roll rate, portray fewer distinct variations across the classes. However, it's noteworthy that lateral acceleration demonstrated a trend consistent with IQR width variations across Class 1, 2, and 3, albeit with less pronounced differences between the classes. The subsequent sections will concentrate on an in-depth examination of Roll Angle and Roll Rate, the two lateral features that showcased the most distinguishable and significant variations across the identified clusters. These features provided rich insights into the diverse vehicle dynamics associated with the different class labels.

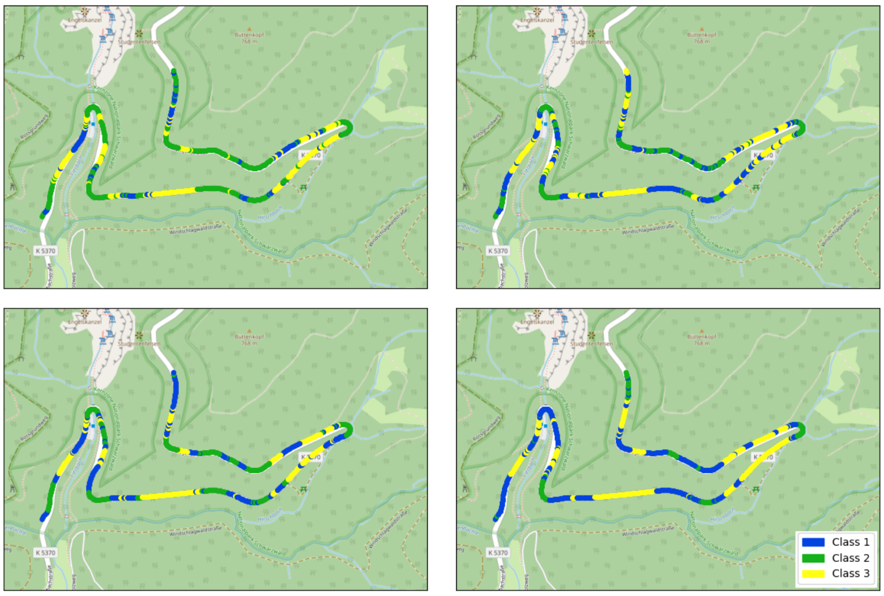


Figure 6.5: GPS visualization of clusters identified using Deep Linear DAE models for rural roads. (a) Top left: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 9, (b) Top right: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 10, (c) Bottom Right: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 6, (d) Bottom Left: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 5

Roll Angle : Analyzing the Roll Angle (RA) characteristics across the identified cluster labels revealed compelling observations. The Interquartile Range (IQR) width in Class 3 mostly appeared to be the smallest among all models, followed by slightly larger widths observed in Class 1. Intriguingly, Class 2 generally exhibited the widest IQR, signifying substantial variability in the Roll Angle dynamics across the dataset, with the exception found in one particular model (AE BN2 with IPL 1.6 s) see (Fig. 6.8), where Class 1 displayed the highest IQR width. Notably, the absence of outliers in the RA box plots belonging to Class 2, except in specific model instances, contrasted with the frequent occurrence of outliers in the RA box plots of Class 1 and Class 3. This aligns with the inference drawn from GPS visualization, where Class 2 prominently corresponds to ride segments characterized by sharp turns. Sharp turns often induce rapid changes in the vehicle's orientation, leading to abrupt variations in the Roll Angle, which substantiates the wider spread observed within Class 2. The absence of outliers, such as those observed in Class 2 for most models (with exceptions in specific model instances), could imply a more uniform or consistent pattern in the RA dynamics within that cluster. Outliers generally represent data points that significantly deviate from the typical behavior observed in a dataset. The lack of outliers in Class 2 indicates that the RA data within this cluster tends to follow a more consistent trend. This consistency in the roll-angle dynamics within Class 2 suggests a more homogenous behavior or driving pattern associated with the segments identified by this cluster label. Therefore, the coherence between the GPS-identified segments with sharp turns and the distinct RA characteristics in Class 2 supports the association between road structure implied by GPS data and the characteristics observed in Class 2.

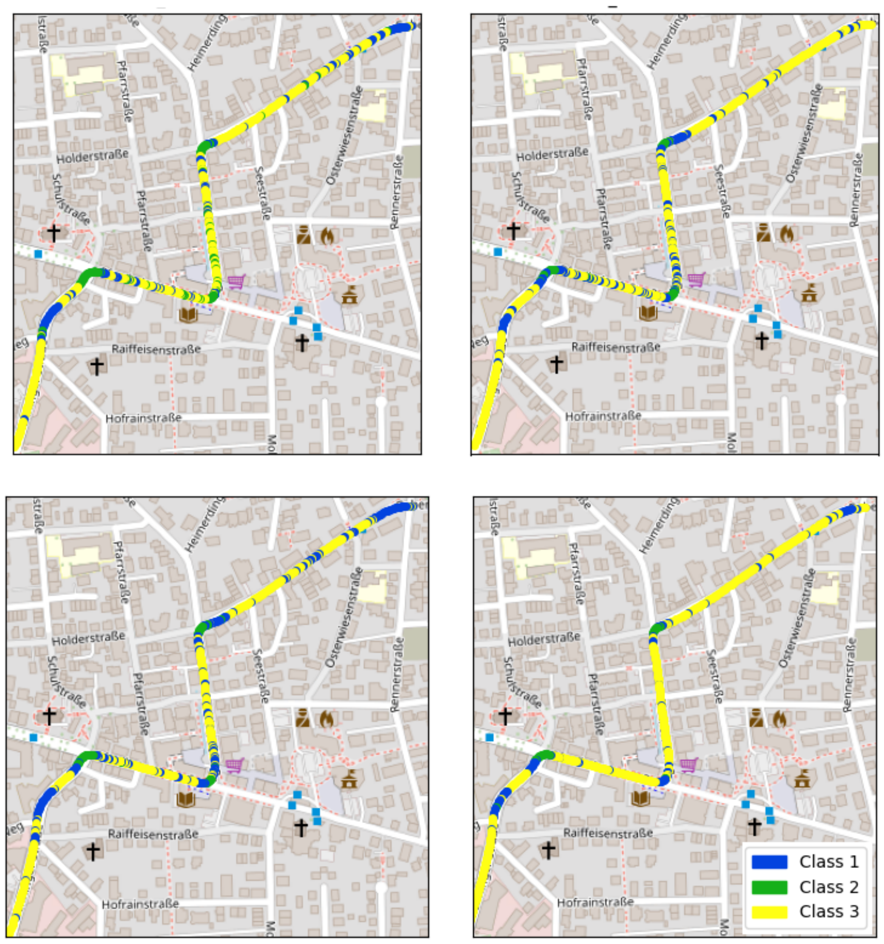


Figure 6.6: GPS visualization of clusters identified using Deep Linear DAE models for urban roads. (a) Top left: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 9, (b) Top right: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 10, (c) Bottom Right: Cluster outputs for DAE BN1 model with IPL 1.0s and LSD 6, (d) Bottom Left: Cluster outputs for DAE BN1 model with IPL 1.6s and LSD 5

Roll Rate : Roll rate signifies the rate of roll angle alteration over time, depicting the speed at which a vehicle leans or tilts. Extreme roll rates emerge during swift maneuvers or abrupt directional shifts, potentially indicating vehicular instability or sudden motions. Understanding a vehicle's dynamic behavior during lateral motion-inducing actions like turns or maneuvers heavily relies on this parameter. Its inclusion in the analysis is critical for comprehensively evaluating distinct clusters formed by the AE variants. Observing Roll Rate box plots across different clusters revealed consistent patterns. Class 3 consistently exhibited the smallest IQR width, representing lower roll rate values. This trend persisted across various IPL values within an AE variant (Fig. 6.9). In contrast, Class 1 consistently displayed the second-highest IQR widths, slightly surpassing Class 3 values. This consistency was prevalent across most model variants, except for AE BN2 with IPL 1.6 s, where Class 1 displayed the highest IQR width. Class 2 consistently presented the highest range of Roll Rate values, signifying the most extreme rates among its time samples.

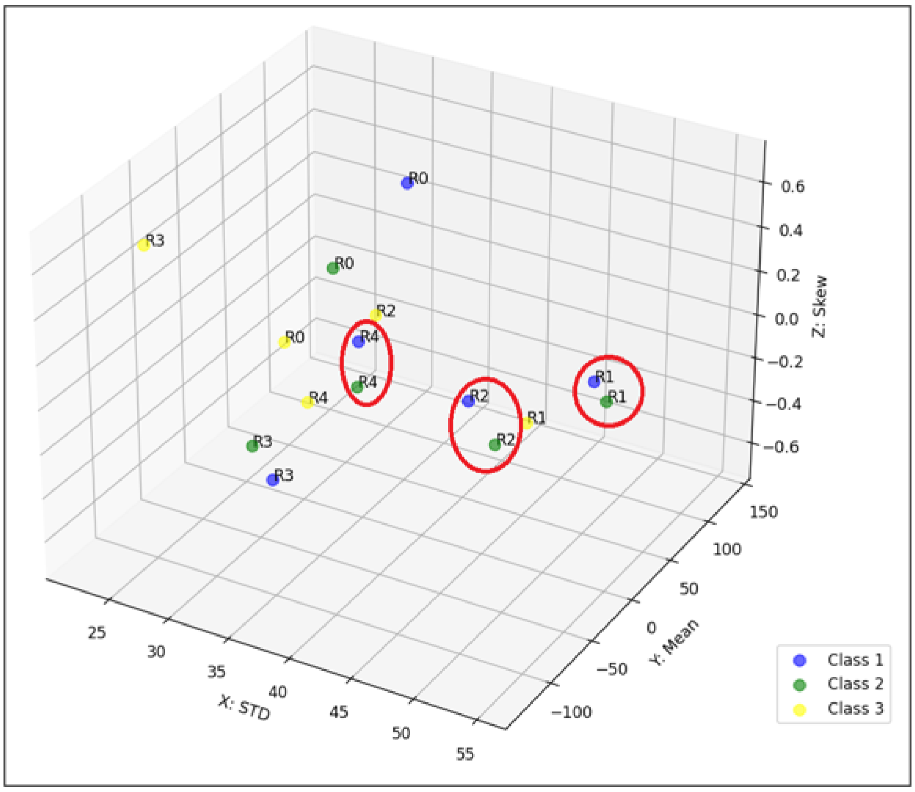


Figure 6.7: Statistical Profile of LS Features from DAE BN2 Model for IPL 1.6 s after Label Switching

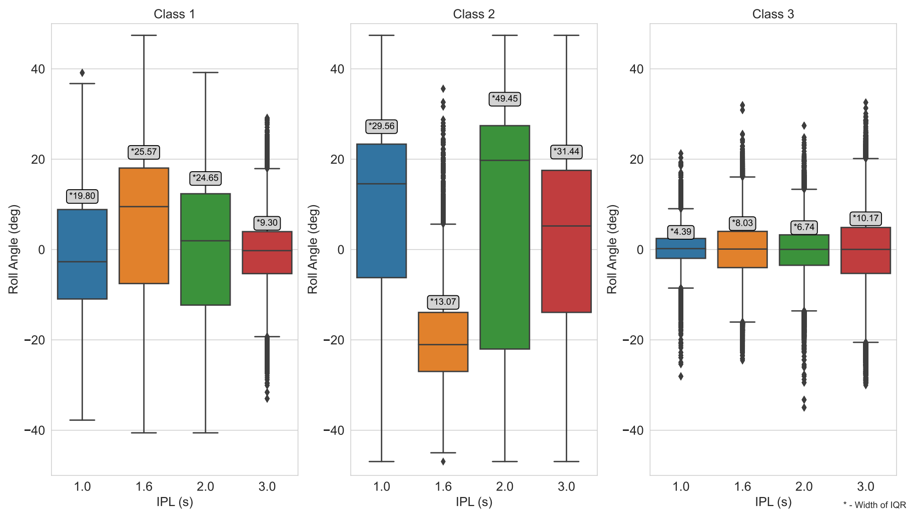


Figure 6.8: Roll Angle Box Plot Analysis within the Class Groups Identified by the LS of DAE BN2 Models

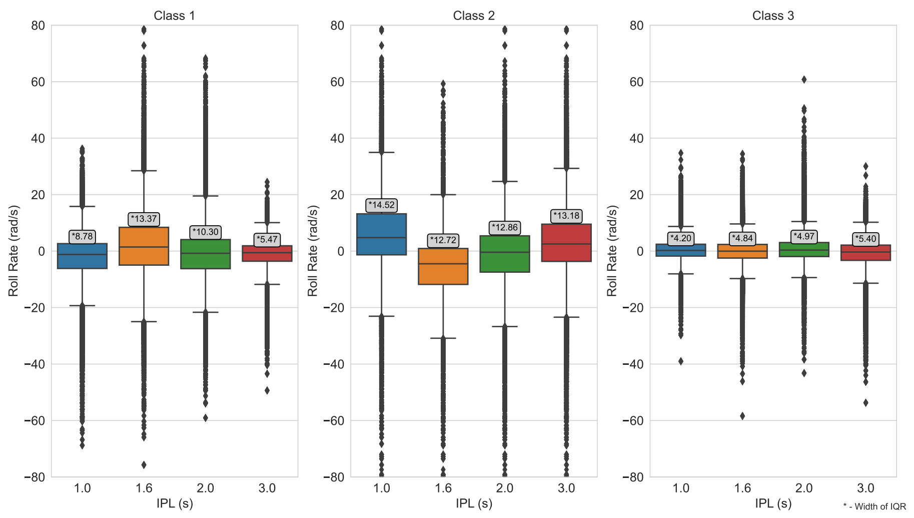


Figure 6.9: Roll Rate Box Plot Analysis within the Class Groups Identified by the LS of DAE BN2 Models

Such findings suggest that Class 2 encapsulates segments featuring dynamic and abrupt vehicle motions, potentially indicating sharp turns, vehicle instability, or sudden directional changes. This observation aligns with the interpretation of Class 2 as associated with dynamic and rapid vehicle dynamic changes.

These findings reinforce the link between Class 2 and rapidly changing vehicle dynamics with significant Roll Rate variations. Conversely, Class 3 appears linked with segments displaying nearly constant vehicle dynamics, reflecting the smallest IQR widths. Meanwhile, Class 1 tends to represent slower changes. Furthermore, the consistent trends across IPL values within most AE variants suggest substantial similarities in vehicle dynamics characterization. Based on the different analyses made on the characteristics exhibited by class labels 1, 2, and 3, an apparent correlation emerges between the clusters' SD calculated over the LS features from the AE models and discernible vehicular dynamics. Elevated SD values in the LS features correspond to swift vehicular alterations, such as sharp turns or lane adjustments, while moderate SD levels indicate moderately evolving dynamics in the original features. Conversely, low SD values are indicative of consistent, straight-line riding. This correlation provides a foundation for the forthcoming sections, aiming to delve deeper into the comprehensive analysis of the identified clusters in relation to vehicle dynamics.

7 Experimental Setup for MTP using Hybrid Model

The Performance of the proposed hybrid model is assessed under two scenarios: (a) An Ideal LSTM classifier that's always correct and (b) A realistic scenario where the classifier suffers errors. This is done to realize the best possible outcomes from the proposed MTP and curated clusters. This two-fold analysis gives a better understanding of the true potential of MTPs while also comprehending their behavior with noisy classifiers.

The GMM dataset is used for MTP and consists of class labels assigned to each of the 200ms data samples and are one hot encoded. This imposes a risk of identifying more than one consistent mode across the prediction horizon. Hence, additional measures have to be taken to define what constitutes a mode in the prediction horizon. The complexity of the experimental process grows from here, which involves identifying the best AE configurations that produce good labels and finally fine-tuning the individual components in the model before testing it on the above-described scenario. Thus, the experimental setup for realizing MTP with the proposed hybrid model is not a straightforward process. The experimentation process is organized as shown in 7.1. The results of fine-tuning are important to test the MTP under scenarios 1 and 2. Hence, the inferences of fine-tuning are discussed in this section. The actual results pertaining to the performance of the proposed MTP realization are detailed in chapter 8.

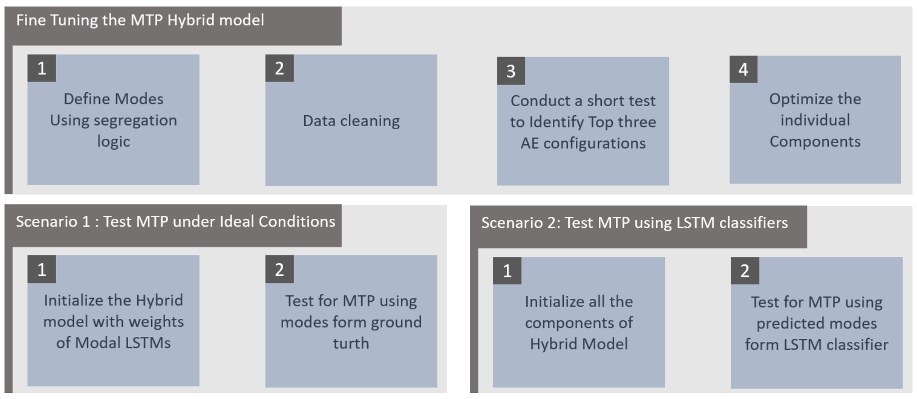

Fine Tuning the MTP Hybrid model
1
Define Modes Using segregation logic
2
Data cleaning
3
Conduct a short test to Identify Top three AE configurations
4
Optimize the individual Components
Scenario 1 : Test MTP under Ideal Conditions
1
Initialize the Hybrid model with weights of Modal LSTMs
2
Test for MTP using modes form ground turth
Scenario 2: Test MTP using LSTM classifiers
1
Initialize all the components of Hybrid Model
2
Test for MTP using predicted modes form LSTM classifier

Figure 7.1: Overview of Experimental Setup

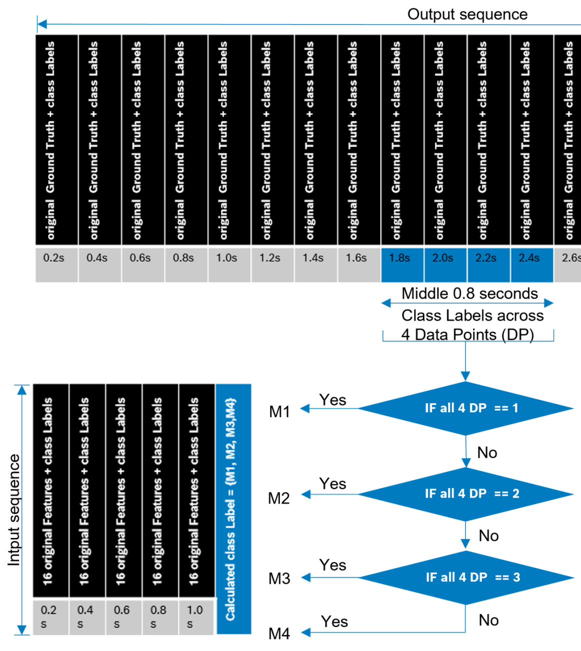


Figure 7.2: Data Segregation Logic

7.1 Defining Modes using Data Segregation Strategy

Utilizing these class labels opens up diverse methodologies for categorizing the dataset. The choice of using the output sequence to formulate the segregation logic stems from the works [DT18][MHZ+21], which demonstrates using future modes to benefit from MTP. A specific number of 0.8 seconds is chosen because of the sporadic occurrence of class labels, which is discussed in detail in the following section. Fig. 7.2 presents a graphical representation illustrating this refined segregation logic. After assigning the input sequence the Modes they belong to M1, M2, M3, M4, the input sequences and their corresponding output sequences are grouped into four datasets, namely DM1, DM2, DM3, DM4, corresponding to times series data belonging to newly calculated modes: M1, M2, M3, and M4 respectively.

Segregating input sequences based on the occurrence of class labels within the middle 0.8 seconds of the corresponding output sequences is pivotal for categorizing them into M1, M2, M3, and M4. Mode M1 consists of data points identified as class 1 in the middle 0.8s. Likewise, M2 and M3 represent time samples from class 2 and class 3, respectively. M4 represents cases where there are more than two class labels within the middle 0.8s. Here on, the LSTM models assigned to classes M1, M2, M3, and M4 are referred to as LSTM1, LSTM2, LSTM3, and LSTM4, respectively, and will collectively be referred to as ‘Modal LSTMs’. LSTM1 is now expected to learn sequences that have moderate variations in their lateral dynamics with more controlled or gradual changes over time. LSTM3 should model the samples that have very small variabilities in their lateral dynamics and exhibit almost no changes over time. In contrast to these models, LSTM2 is anticipated to characterize time sequences where there are high variations in the lateral dynamics with rapid temporal changes.

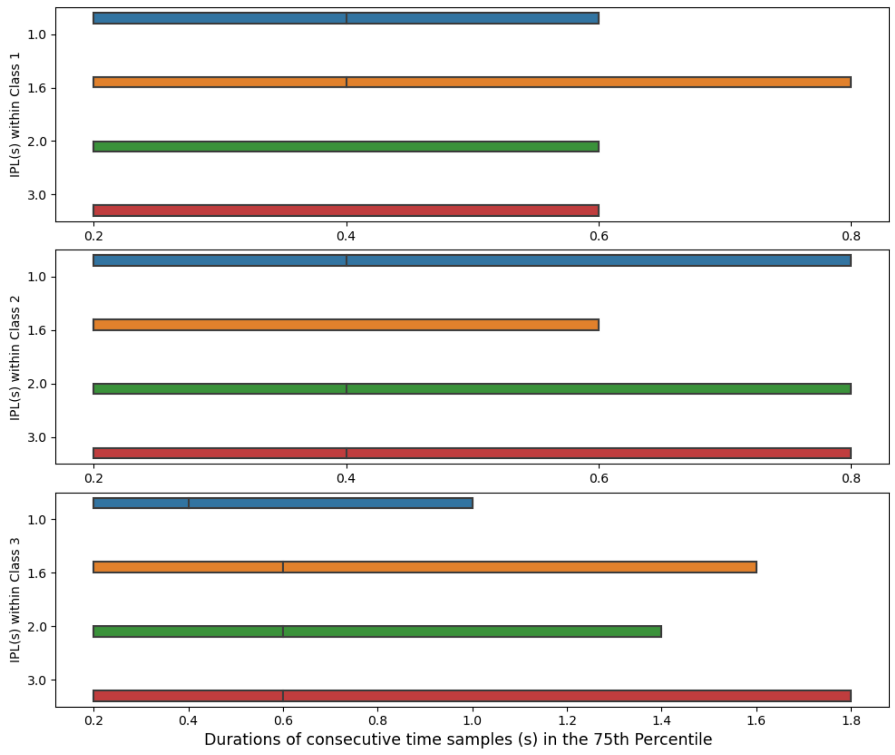


Figure 7.3: Distribution of Class Label Time Durations in the 75th Percentile Before Cleaning

7.2 Data Cleaning with Controlled Interpolation Algorithm

The segmentation strategy directly affects the capacity of the LSTMs and, hence, is fundamental to our approach to achieving MTP. However, the identified class labels exhibit sporadic spikes or short occurrences of classes within intervals such as 0.2s or 0.4s. These abrupt occurrences pose a challenge as they may hinder the balanced segregation of input sequences into their respective classes. These spikes could potentially increase the assignment to M4, even when the time sequence under examination predominantly consists of consecutive occurrences of data points belonging to a specific class, thereby increasing the number of samples in M4. To demonstrate the distribution and effect of this sporadic nature in the entire dataset, a box plot is created. Since these sporadic spikes only occur in the lower 25th percentile of the Box plots, and to ease visibility, the visualization is restricted to show only the 75th percentile of the dataset (Larger time durations are beneficial; hence, the time durations of the upper 25% of the class labels are not shown). Fig. 7.3 shows the 75th percentile of the time durations of the identified classes.

The visualization of the time durations of each class unveils an important insight. The occurrences of sporadic spikes lead to major shifts in the time durations of the occurrence of class labels. Even though the box plot is calculated for the 75th percentile, observing the distributions across all input sequence lengths for DAE BN1, the Q1 and Q2 regions are overlapped, meaning at least 25% of the class occurrences lasts for an average of 0.4 seconds. What this means for the data segregation logic depicted in Fig. 7.2 is that almost 25% of the time series sequences after sampling will mostly be assigned to M4, and this impact is influenced just by the 25th percentile. Moreover, the time

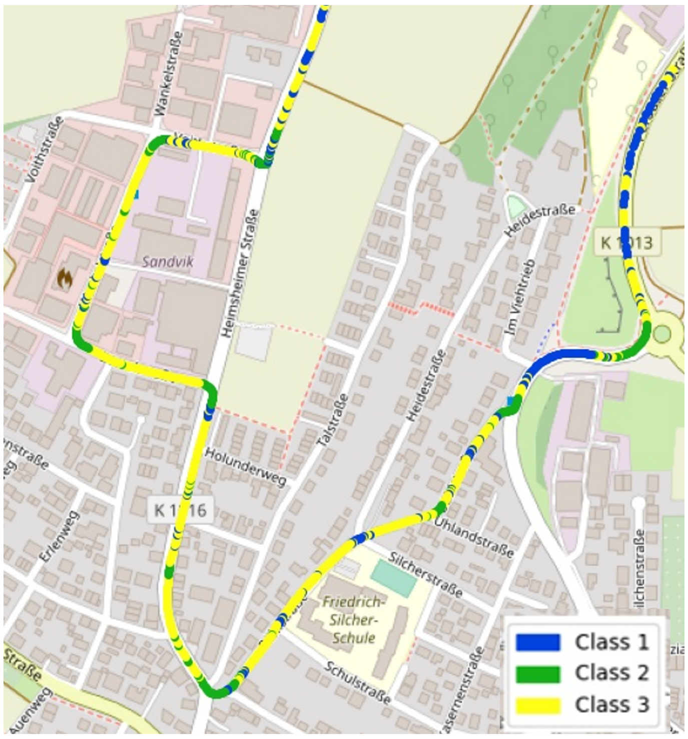


Figure 7.4: GPS Visualization of Class Labels before Cleaning

durations of class 1 across all but 1.6s IPL have the whole 75th percentile class occurrences lasting for a very short 0.6s, and all of these sequences, when sampled for dataset creation, will mostly be assigned to M4.

Ensuring a more balanced categorization demands addressing the short sporadic occurrences of samples belonging to a particular class. To tackle this issue effectively, a custom- algorithm is introduced in the following sub-section (Algo. **??**). This algorithm aims to refine the data by smoothing out these short-lived spikes. The controlled interpolation algorithms aim to achieve the least information loss by considering the surrounding context before smoothing the sporadic occurrences.

Controlled Interpolation Algorithm (CIA)

The CIA Algo . 7.1, in combination with the context-based imputation algorithm (See Algo . 7.1), is designed to process and clean sporadic occurrences of class label values. This algorithm estimates and reassigns the sporadic durations within each ride file by leveraging the temporal class label context.

The algorithm identifies sporadic durations of 0.2s and 0.4s from the dataset, considering these time intervals as candidates requiring cleaning and restoration (Fig.7.5). These durations are targeted due to their intermittent nature, indicating irregularities in the temporal sequence of class labels. (See Fig.7.3) After identifying the desired sporadic durations to eliminate, the cleaning begins by first reassigning the shortest ones. The main characteristic of this algorithm is that it

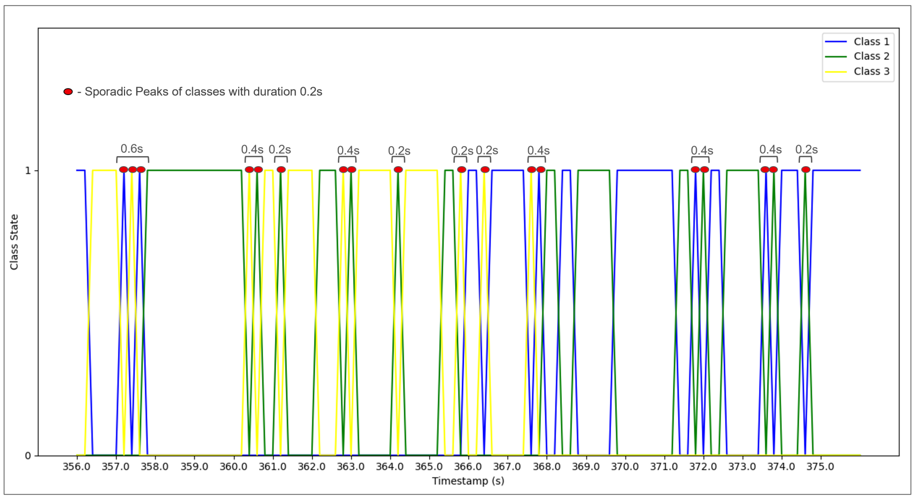


Figure 7.5: Step 1 of CIA. Identify sporadic candidates for cleaning

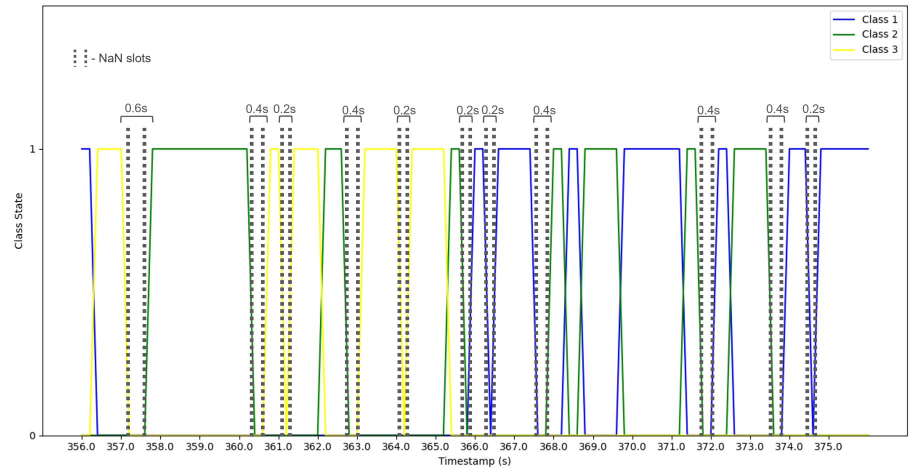


Figure 7.6: Step 2 of CIA. Remove the sporadic class labels assigned to data points, and fill them with NaN values

gradually reassigns the chosen sporadic durations instead of cleaning all durations at once. The class labels of the shortest durations are first assigned with NaN values Fig.7.6. These short bursts of NaN values are then subjected to context-based imputations. The context-based imputations fill in the NaN values based on the classes preceding and succeeding the NaN slots. When the preceding and succeeding class labels are different from each other, the first half of the NaN slots are filled with the preceding class, and the last half of the solutions are filled with the succeeding class label (Fig.7.7). Doing so ensures less information loss and accounts for a more controlled imputation. The details of the Context-Based Imputation algorithm are available in Algo.7.1. This controlled reassignment approach is continued for the remaining sporadic durations of interest, and the cleaned time series of the example considered in Fig.7.5 is shown in Fig.7.8. The complete details of the custom interpolation algorithm are detailed in Algo.7.1, Algo.7.2 and Algo.7.3

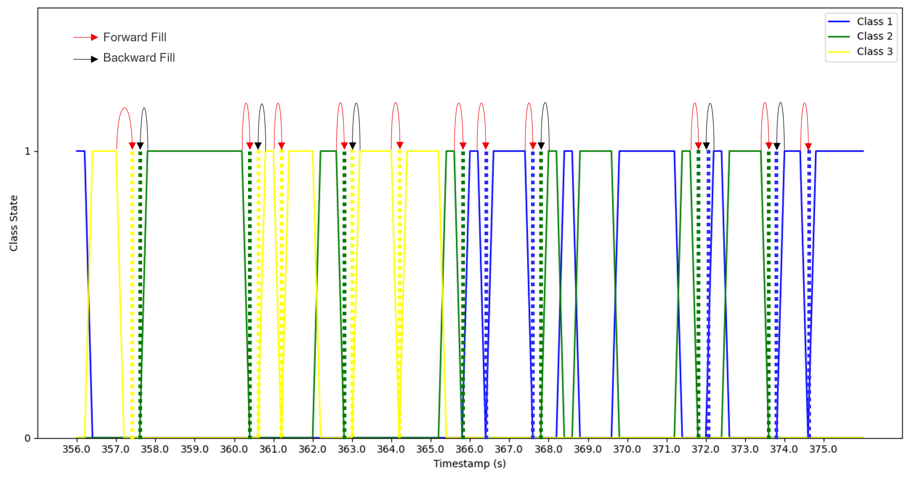


Figure 7.7: Step 3 of CIA. Fill in the NaN slots based on the surrounding context. Use the preceding class label to forward fill and the succeeding class to backward fill the NaN slots

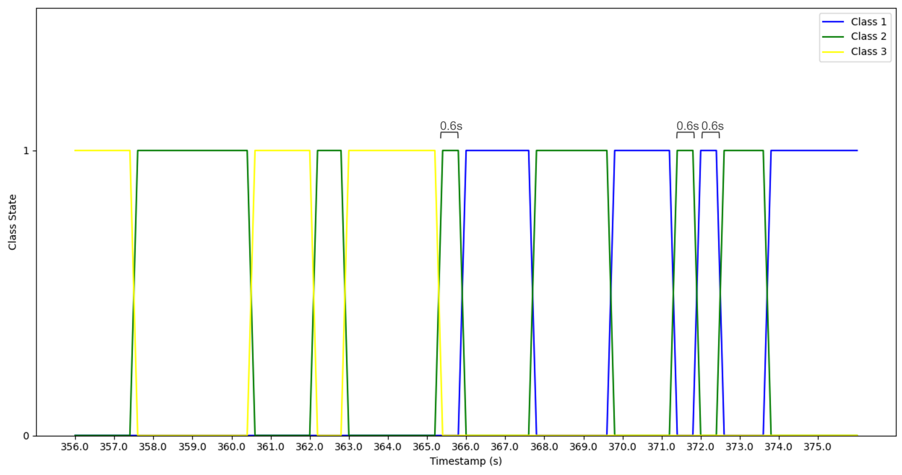


Figure 7.8: Durations of Class Labels After Cleaning Sporadic Classes

Algorithm 7.1 Custom Interpolation Algorithm

```
function CustomInterpolation
    Input: RideFile
    ClassDuration ← ComputeClassDurations(RideFile)
    DurationsToClean ← [0.2s, 0.4s]
    for Clean in DurationsToClean do
        RowsToClean ← ClassDuration['Durations' == Clean]
        FillNaNValues(ClassDuration, RowsToClean)
    end for
    Return: Cleaned ClassDuration
end function
```

Algorithm 7.2 FillNanValues

```
function FillNaNValues(ClassDuration, RowsToClean)
    Input: ClassDuration, RowsToClean
    Initialize RowsToFill, CountConsecutiveNaN, CR, Start, Stop, PrecedingClass, SucceedingClass, Slice
    Initialize an empty list RowsToFill, a counter CountConsecutiveNaN to 1
    CR ← 0, Start ← 0, Stop ← 0
    PrecedingClass ← Null, SucceedingClass ← Null
    while Not all elements in RowsToClean are visited do
        CR ← RowsToClean[CountConsecutiveNaN]
        Append RowsToFill with CR
        while RowsToClean[CountConsecutiveNaN] == CR + 1 do
            Increment CountConsecutiveNaN
            Append RowsToFill with RowsToClean[CountConsecutiveNaN]
        end while
        Remove duplicates from RowsToFill and sort it
        if CountConsecutiveNaN == 1 then
            if PrecedingClass ≠ Null then
                ClassDuration[CR] ← ClassDuration[RowsToClean[CountConsecutiveNaN]]
            end if
        else
            Slice ← CountConsecutiveNaN/2
            if CountConsecutiveNaN is odd then
                Increment Slice
            end if
            for index in range(0, Slice) do
                ClassDuration[RowsToFill[index]] ← PrecedingClass
            end for
            for index in range(Slice, len(RowsToFill)) do
                ClassDuration[RowsToFill[index]] ← SucceedingClass
            end for
        end if
        Reset CountConsecutiveNaN to 1 and empty the RowsToFill list
    end while
    Return: Updated ClassDuration
end function
```

Algorithm 7.3 calculateClassDurations

```
function ComputeClassDurations(RideFile)
    Input: RideFile
    Initialize ClassTimeStamps, TimeDurations, ObservedClass, CountTimeStamp, Start, Stop
    Add calculated class labels to each ride file
    $N \leftarrow$ total number of time samples or data points sampled at 200ms in a ride file
    Initialize ClassTimeStamps dataframe with $N$ rows and two columns: TimeStamps, ClassLabels
    Create a new 1D array called TimeDurations with $N$ number of rows, initialized with zeros
    ObservedClass $\leftarrow$ first class label in the ClassTimeStamps dataframe
    CountTimeStamp $\leftarrow$ 1, Start $\leftarrow$ 0, Stop $\leftarrow$ 1
    for row in range(2, $N$) do
        if current class label == ObservedClass then
            Increment CountTimeStamp
        else
            Update Stop with the current row index
            for index in range(Start, Stop) do
                Update TimeDurations with CountTimeStamp for indices from Start to Stop
            end for
            Update Start with the current row index + 1
            Reset CountTimeStamp to 1
            Update ObservedClass with the class label in the current iteration
        end if
    end for
    Multiply integer values in TimeDurations array by 0.2 to represent durations in seconds
    Create a new dataframe ClassDuration with $N$ rows and 3 columns: TimeStamps, ClassLabels,
Durations
    Assign values from ClassTimeStamps to ClassDuration for the first two columns, and assign
TimeDurations to the third column
    Return: ClassDuration dataframe
end function
```

By employing this customized interpolation algorithm, we are able to create a more coherent and consistent representation of class labels within the dataset, and this serves as an attempt to prevent too many samples from being assigned within M4. Fig.7.9 the box plot used to visualize the distribution of class durations across the class labels generated by the AE BN1 models. The plot shows a significant change in the data distribution trend. The class durations observed in the first quartile now last longer than the candidate sporadic durations chosen for learning.

It should also be noted that just by clearing two of the shortest sporadic durations, the class durations have grown in length, and this is very evident from smaller Q3 values observed in Fig.7.3 compared to the longer ones in Fig.7.9 The effects of cleaning up the sporadic class spikes lasting for 0.2s and 0.4s can be visualized with GPS information and a sample visualization a small part of ride file is shown in Fig.7.10.

Algorithm 7.1 Context Based Imputation

```
function ContextBasedImputation(ClassDuration, RowsToClean)
    Input: ClassDuration, RowsToClean
    Initialize RowsToFill, CountConsecutiveNaN, CR, Start, Stop, PrecedingClass,
SucceedingClass, Slice
    Initialize an empty list RowsToFill, a counter CountConsecutiveNaN to 1
    CR ← 0, Start ← 0, Stop ← 0
    PrecedingClass ← Null, SucceedingClass ← Null
    while Not all elements in RowsToClean are visited do
        CR ← RowsToClean[CountConsecutiveNaN]
        Append RowsToFill with CR
        while RowsToClean[CountConsecutiveNaN] == CR + 1 do
            Increment CountConsecutiveNaN
            Append RowsToFill with RowsToClean[CountConsecutiveNaN]
        end while
        Remove duplicates from RowsToFill and sort it
        if CountConsecutiveNaN == 1 then
            if PrecedingClass ≠ Null then
                ClassDuration[CR] ← ClassDuration[RowsToClean[CountConsecutiveNaN]]
            end if
        else
            Slice ← CountConsecutiveNaN/2
            if CountConsecutiveNaN is odd then
                Increment Slice
            end if
            for index in range(0, Slice) do
                ClassDuration[RowsToFill[index]] ← PrecedingClass
            end for
            for index in range(Slice, len(RowsToFill)) do
                ClassDuration[RowsToFill[index]] ← SucceedingClass
            end for
        end if
        Reset CountConsecutiveNaN to 1 and empty the RowsToFill list
    end while
    Return: Updated ClassDuration
end function
```

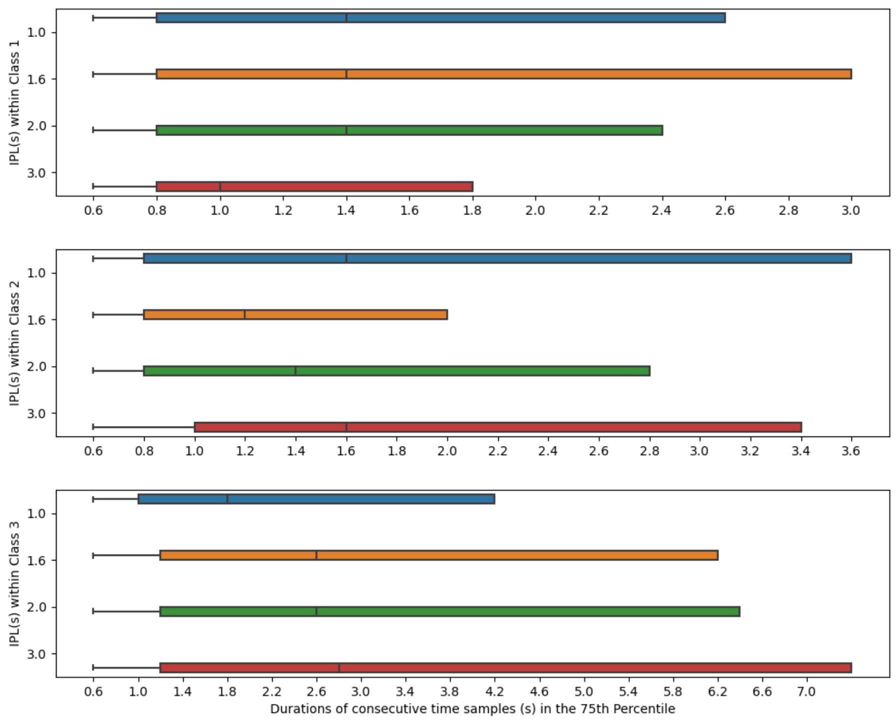


Figure 7.9: Distribution of class label time durations in the 75th percentile after cleaning. After applying CIA there is an increase in the overall all class durations as evident from the width of IQR

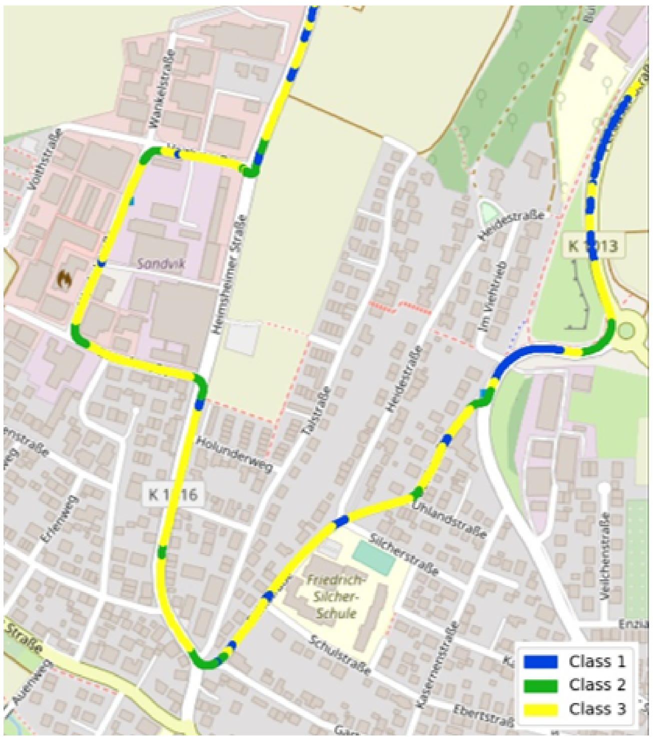


Figure 7.10: GPS Visualization of Class Labels After Cleaning

Layer ID	Input Layer	Hidden size	Output Layer
1	I	H	H
2	H	–	H
3	H	–	$H/2$
3	$H/2$	–	4

Table 7.1: Layer Proportions of the LSTM Classifier Model

7.3 Test to Identify the Top Three AE Configurations

The objective of the classifier is to predict the future mode of the trajectories provided an input sequence of length 1.0 seconds. Since we have experimented with different versions of the baseline model, the predictive classifier is based on a single LSTM unit. This classifier takes in an input sequence and gives out the probabilities of the previously identified four modes. The mode with the maximum probability is then used to choose the modal LSTM and make roll angle predictions. The architecture of the proposed model is very similar to the modal LSTM unit. The key difference to modal LSTM arises in the 4th linear layer, which only has four nodes, each to denote the probability of occurrence of a mode. The corresponding layer proportions of the LSTM classifier are shown in Tab.7.1

After clusters were identified within the dataset, it became necessary to evaluate the feasibility of employing MTP by leveraging the clusters as modes. Despite the clusters proving informative, there are inherent challenges in constructing a reliable classifier to predict future modes. Traditionally, MTP embraces a probabilistic approach, encapsulating uncertainty by offering probabilistic trajectories rather than a single, deterministic prediction. However, the data segregation logic provides a different grouping of data for each type of the previously identified AE configuration. Hence, an initial exploration involves evaluating the twenty-four different variants of the proposed BN1 and BN2 AE models to identify optimal LS representation followed by GMM clustering. It is now essential to narrow down the selection to at least three specific AE configurations that best suit the task at hand.

The following demonstration emphasizes the utility of the labels by assuming an ideal deterministic classifier that predicts the future modes with 100% probability. This experimentation is done without using the classifier to understand how MTP works in an ideal setting and to analyze the effectiveness of the identified class labels without incurring errors from the classifier. This assumption is realized by utilizing the ground truth labels obtained through the segregation logic discussed in the previous sections; the momentary solution is to select a single pre-trained model from a set comprising LSTM1, LSTM2, LSTM3, and LSTM4. The predicted outcomes of such a selection depend on the segregation logic and the effectiveness of the class labels at each data point. Hence, this approach is a simple yet effective way to determine the most suitable labels to design and optimize the predictive classifier. Following this assumption, an experiment is conducted to identify the top three AE configurations that produce good clustering. After this step, the best-identified models are optimized.

7.3.1 Experimental Setup to Identify Top Three AE Models

For the following demonstration, all the dedicated LSTMs are given an input sequence spanning one second. The sampling rate is maintained at 200ms, and the segregation logic used to identify the four different modes is directly employed for selecting the LSTM associated with the future mode.

Training, Validation

Since we have to assess all 24 different configurations of the AE model, the training loop for pretraining the sub-models was set to a short 20 epochs. The optimization of each sub-LSTM is essential but is only carried out for the top three best-performing models. To ensure that there is a fair comparison for all the sub-models within the 24 different configurations, the same LSTM architecture is used for all the sub-models. To ensure that a good configuration is used for this comparison, we take the hyperparameters of the baseline LSTM architecture. The baseline model has a single LSTM unit, with a hidden size of 64, that is capable of generalizing over all the modes. Following the LSTM layer is one fully connected linear layer and a final transformation layer.

Evaluation Metric : The Root Mean Squared Error (RMSE) is used to evaluate the roll angle predictions, while the MSE and R2 are used to train and validate the sub-LSTM predictions. Just as MSE, RMSE encapsulates the sensitivity to errors' magnitude and direction, but it presents the assessment in a more interpretable scale by considering the square root of the average squared differences. The formulation of RMSE is shown in Eq. 7.1. This characteristic becomes particularly pertinent in the context of time series predictions, providing a comprehensible measure of deviation from the true values.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}} \tag{7.1}$$

The training and validation losses for the sub-models are not displayed in detail since no significant insights were observed. Instead, the selection is completely based on the observed overall RMSE values from the testing phase. The train, test, and validation datasets used throughout the 24 different configurations are the same. However, it should be noted that due to the segregation logic that is based on the class labels Provided by specific AE configurations, the sub-datasets used for training and validating each LSTM model are different across the 24 configurations.

7.3.2 Results and Inference

It should be noted that since the IPL used for each configuration is different, there are very small variations in the test samples. The results for the described test are shown in Tables. 7.2, 7.3 and 7.4

From the short test conducted for identifying the top three models, one general observation is that, by assuming the labels generated, the overall RMSE for roll angle predictions is better than the overall RMSE observed under unimodal trajectory prediction. Another interesting finding is that the

IPL (s)	BN1	BN2
1.0	7.312	6.917
1.6	7.2863	6.850
2.0	7.4954	7.3392
3.0	7.3241	7.4693

Table 7.2: Summary of the overall RMSE of roll angle (deg) prediction results from the hybrid model using DAE labels

IPL (s)	BN1	BN2
1.0	7.1853	7.3271
1.6	7.0210	6.9247
2.0	7.6813	7.5257
3.0	7.3569	7.4136

Table 7.3: Summary of the overall RMSE of roll angle (deg) prediction results from the hybrid model using LSTMAE labels

top three best-predicting models had a higher reconstruction error compared to their counterparts, and the summary of observations is stated in Tab.7.5. For the other AE configurations. The BN2 versions had a comparable prediction and sometimes even outperformed their corresponding BN2 variants. One key takeaway from the label identification tests is that the LSD associated with the top three models are the three least values we had earlier described in the hyperparameter optimizations space (5, 6, 7). Combining this inference with the results of the ablation study conducted to find the best combination of features to send along with LS for GMM clustering, we can say that a good, condensed LS with minimal combinations provides a balance between the insights offered by the yaw rate and pitch rate. When the pitch and the yaw rate are combined with an LS, it seems to make the problem of identifying GMM components cumbersome.

IPL (s)	BN1	BN2
1.0	7.6045	7.5178
1.6	7.3108	7.5058
2.0	7.4132	7.6135
3.0	7.0479	7.5640

Table 7.4: Summary of the overall RMSE of roll angle (deg) prediction results from the hybrid model using BDLSTMAE labels

IPL (s)	**AE Configuration**	**LSD**	**Input Reconstruction error MSE**	**Overall RMSE of Roll Angle (deg)**
DAE BN2	1.6s	5	0.1981	6.850
DAE BN2	1.0s	6	0.1163	6.917
LSTMAE BN2	1.6s	6	0.3298	6.9247

Table 7.5: Top three best AE configurations for generating LS

7.4 Training Individual Components in Hybrid Model

To assess the performance of MTP in both (a) and (b), The Modal LSTM units first have to be optimized and trained over their respective sub-datasets. Then following this, the LSTM classifier is trained over the GMM datasets generated by the top 3 AE configurations. From here on, the DAE models with IPL of 1.0s and 1.6s are referred to as Configuration 1 and Configuration 2, and the LSTM AE BN2 with IPL of 1.6s are referred to as Configuration 3. Three separate instances of the LSTM classifier are realized to classify the modes identified by configurations 1, 2, and 3, and these classifiers are referred to as LSTM Classifier1 (LC1), LSTM Classifier2 (LC2), and LSTM Classifier3 (LC3), respectively.

7.4.1 Data Preparation

The segregation logic is applied over the class labels that are available for each point in the ground truth (GMM dataset). This new information is added to the ground truth sequence, so each output sequence has additional labels $M \in \{1, 2, 3, 4\}$, where each element in M denotes the corresponding mode. After adding the mode labels to the GMM dataset, the train, test, and validation sets are further split into four categories each.

7.4.2 Evaluation Metrics:

Similar to the previous experimental setups, the MSE is used as the cost function for training each of the optimized modal LSTM units. The roll angle predictions for both cases (a) and (b) are evaluated over the overall RMSE value. In addition to these metrics, the cross entropy loss function is used to train the LSTM classifier.

Cross Entropy Loss (CC) is used to evaluate the distinction between two probability distributions. In terms of information theory, CC pertains to the quantification of information, which signifies the number of bits essential for encoding and transmitting an event [Mur12]. CC is commonly used as a loss function in multiclass classification problems. The mathematical equation is given in Eq. 7.2. $\mathrm{H(t, p)}$ represents the cross-entropy between the true distribution t and the predicted distribution p.
)

Hyper parameters	**LSTM1**	**LSTM2**	**LSTM3**	**LSTM4**
Hidden State	256	256	256	256
Learning Rate	0.0006	0.0008	0.0007	0.0008
Dropout rate	0.0581	0.1366	0.1258	0.1138
Objective Value	50.1161	102.9659	12.6489	59.95

Table 7.6: Hyper parameters of the Modal LSTM units in configuration 1

Hyper parameters	**LSTM1**	**LSTM2**	**LSTM3**	**LSTM4**
Hidden State	256	256	256	256
Learning Rate	0.0001	0.0009	0.0006	0.001
Dropout rate	0.07359	0.0370	0.0642	0.064
Objective Value	590.0957	49.798	18.7414	69.59

Table 7.7: Hyper parameters of the Modal LSTM units in configuration 2

7.4.3 Modal LSTM Optimization for Top Three Best AE Configurations

Optuna trials were conducted for ten epochs for each of the modal LSTM units in the top three best-performing models. The train and validation set of each sub-dataset were chosen for this optimization problem; this ensures that the LSTM unit is fine-tuned for the dataset it is learning on. Similar to the optimizations carried out for the reconstruction process, the MSE is used as the loss function during training and is also chosen as the objective loss function (to decide on the favorable validation loss). Unlike the reconstruction process, the number of hyperparameters to fine-tune is rather small due to the existence of a baseline model. Tables 7.6, 7.7, and 7.8 show an overview of the chosen hyperparameters from the Optuna trials. The objective value signifies the MSE value calculated over the validation set during the optimization process.

$$\mathrm{H(t,p)} = -\sum_{s\in S} t(s) \cdot \log(p(s)) \tag{7.2}$$

Hyper parameters	**LSTM1**	**LSTM2**	**LSTM3**	**LSTM4**
Hidden State	256	256	256	256
Learning Rate	0.0007	0.0004	0.0005	0.0005
Dropout rate	0.0519	0.0599	0.05047	0.0702
Objective Value	29.7019	163.9068	28.4297	66.807

Table 7.8: Hyper parameters of the Modal LSTM units trained over modes identified fom configuration 3

Training and Validation Based on a few Optuna trials, the LSTM classifier is designed to have a hidden size of 64, and The output size of the classifier is set to 4. A learning rate of 0.001 and a dropout rate equal to 0.115 are selected for model training. After optimizing both the LSTM classifier and the modal LSTMs, they are subjected to training independently since an end-to-end training approach is not used. The training phase is conducted for a short ten epochs for both the LSTM classifier and the modal LSTMs.

7.5 Experimental Setup for Testing MTP under Different Scenarios

After training the individual components of the MTP hybrid model, the testing process involves initializing the MTP hybrid model with each component's weight. The difference in testing for cases a and b results in the use of the predictive classifier. For testing the ideal performance of the MTP hybrid model, the mode labels that are available in the ground truths are directly used for selecting the predictions associated with the future modes. Whereas for scenario b, the trained LSTM classifier selects the future mode based on the highest probability it observes across the different future modes.

8 Results and Analysis

8.1 MTP under Ideal Conditions

In the ideal scenario, the MTP hybrid model demonstrated overall RMSE values that were consistently close across all three classifiers. This similarity indicates that utilizing the modes identified via the top three AE configurations significantly contributes to the effectiveness of the MTP model.

After optimizing the sub-LSTM models, there is a slight difference in the ranking of the top three models. DAE BN2 with an IPL of 1.0s showed the least overall RMSE; this is followed by DAE BN2 with an IPL of 1.6s, and LSTMAE BN2 with an IPL of 1.6s remains unchanged as the third best model. Nevertheless all three configurations offer An overall RMSE not far off from each other. The summary of the best three model's architecture and performance after optimization are shown in 8.1.

Mode 2 samples are the hardest to identify since they exhibit high variability and extreme, rapid changes to the roll angle. Hence, it is worthwhile to illustrate the roll angle predictions for this mode. As illustrated in Fig.8.1 Fig. 8.2, LSTM 2 Fig.8.3 captures the underlying trend in the ground truth as well as has a smaller difference from the actual ground truth. The third best model on the other hand (Fig.8.3) is closer to the ground truth but struggles to follow the trend(Note: these output sequence predictions are taken at random from the test dataset, so there might be other good samples of mode 2 to depict the capacity of AELSTM BN2 with IPL 1.6s).

Analyzing the Classes Identified by the Best Model Candidates

To understand the characteristics of the different modes identified by the AE configurations, it's important to take note of their roll angle and roll rate values distributed within their individual classes calculated via GMM. Fig. 6.8 shows the data distribution of the Roll Angle. From this illustration, it can be observed that most of the data samples in the Q2, Q3, and Q4 regions of the class 2 box plot belonging to the DEA model with IPL of 1.0s is largely concentrated in the

IPL	**AE**	**LSD**	**Input**	**Overall RMSE**
	Configuration		**Reconstruction error**	**of Roll Angle**
(s)			**MSE**	**(deg)**
DAE BN2	1.0s	6	0.1163	6.7908
DAE BN2	1.6s	5	0.1981	6.8249
LSTMAE BN2	1.6s	6	0.3298	6.8819

Table 8.1: Top three best AE configurations for generating LS after optimization

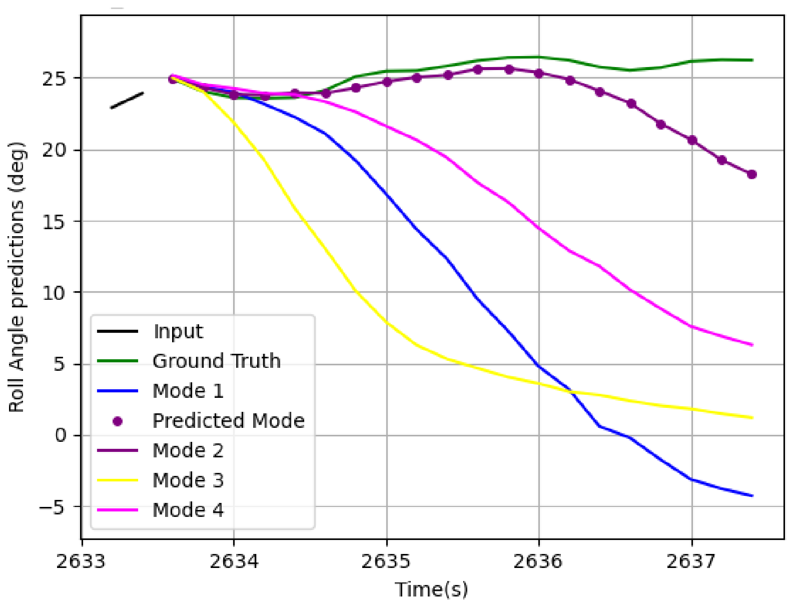


Figure 8.1: Mode 2 predictions for DAE BN2 model with IPL 1.0s under perfect classifier assumption

positive range of roll angle. Referring to the definition of maneuver segments based on lateral dynamics [SWS23] defined in the existing work, this specific case of positive roll angle would mean that class 2 contains more than 50% of data points that are identified to be associated with right turns. Conversely, the Class 2 segments belonging to the second-best model show that all the 75th percentile of roll angles in the box plots falls in the negative range; this would mean that Class 2 of the second-best model has learned to identify roll angles associated with left turns.

Similar behavior in the sign of roll angle distribution is observed in the roll rate across the classes. Class two of the second-best model almost encapsulates all the roll rate samples within the negative roll rate range. This behavior, in combination with the negative roll angle behavior in class two, means that the class 2 samples of the DAE model with IPL of 1.6s are associated with left-turn events and maneuvers. Extending a similar analysis to the class 2 labels of DAE with IPL 1.0s, the upper 75% of the data in the positive scale of roll angle and roll rate denotes that class 2 of IPL 1.0s is most associated with turn scenarios. Contrary to the 1st and the 2nd best model LSTMAE BN2 with IPL 1.6s, the indication as to whether the data samples are associated with left or right turn events is not apparent. The 3rd best model encapsulates both left and right-turning events.

8.2 LSTM Classifier Results

Classification Results : The performance of the classifier is evaluated based on the classification report available from the sci-kit learn library, and the results from the classification for the two best models are shown in Tab.8.3 and Tab.8.4.

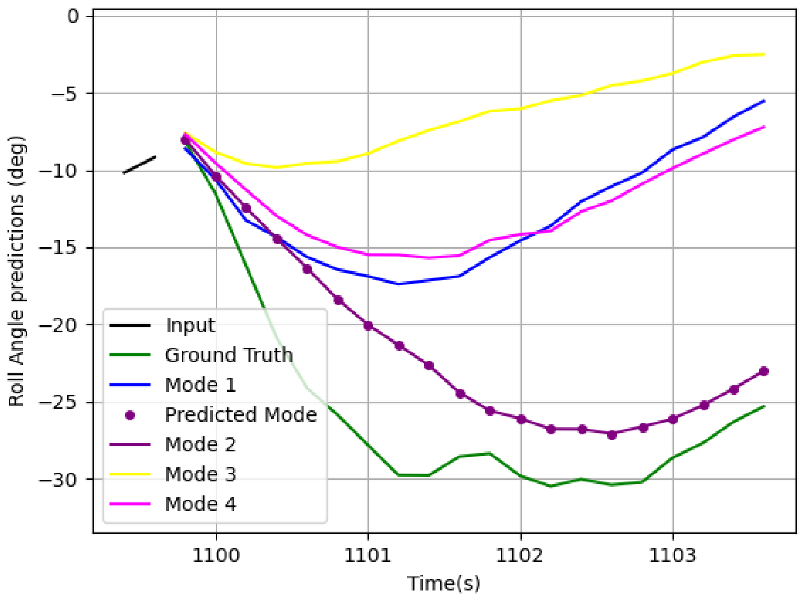


Figure 8.2: Mode 2 predictions for DAE BN2 model with IPL 1.6s under perfect classifier assumption

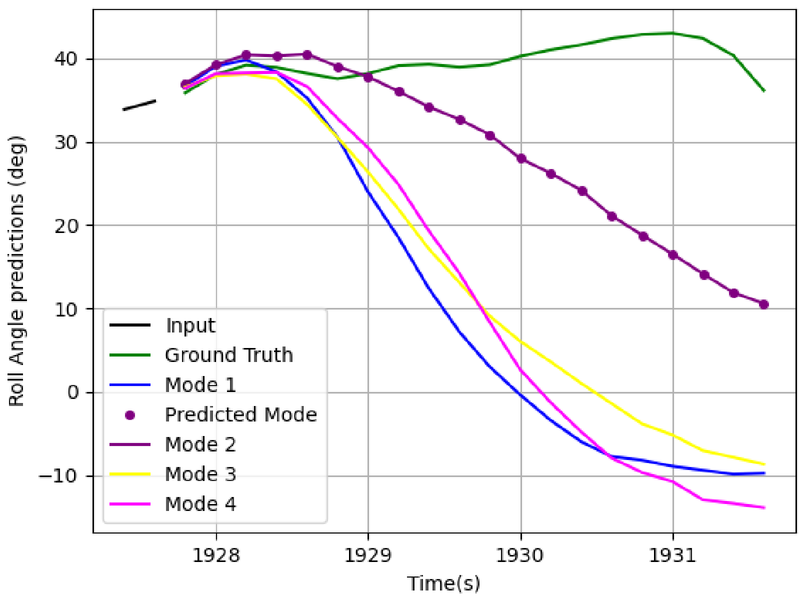


Figure 8.3: Mode 2 predictions for LSTMAE BN2 model with IPL 1.6s under perfect classifier assumption

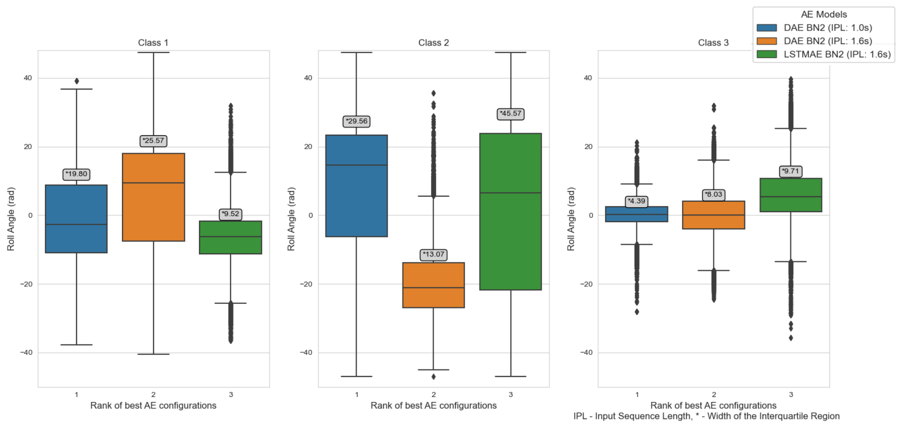


Figure 8.4: Roll Angle distribution within each class of the Top three AE models.

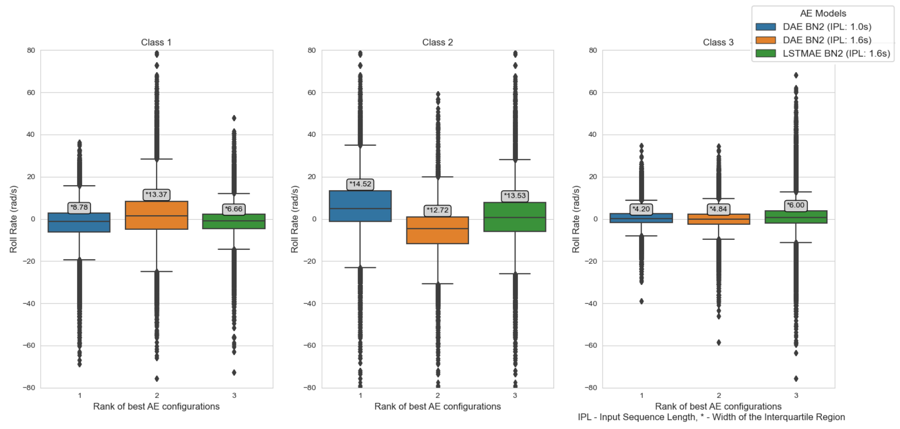


Figure 8.5: Roll rate distribution within each class of the Top three AE models.

Upon a deeper analysis of the classification reports for LC1, LC2, and LC3, several insightful observations emerge. LC1 (trained on modes from DAE BN2 IPL 1.0s) showcases commendable precision (55%) and high recall (70%) for Mode 1, highlighting its capability to identify instances within this mode accurately. This is particularly useful in cases where there are Mode 1 (roll angles with intermediate SD) trajectories. However, LC1 notably struggles with Mode 4, exhibiting low precision (34%) and virtually no recall (1%), demonstrating significant difficulty in correctly predicting instances within this Mode. LC2 (trained on modes from DAE BN2 IPL 1.6) demonstrates superior precision and recall for Mode 3 (72%) compared to the other models. Mode 3 is associated with straight-ride segments in the future (approximately 1.6 seconds into the future), indicating its proficiency in this category. Yet, similar to LC1, it encounters challenges in Mode 4. On the other hand, LC3 exhibits consistent performance in predicting Modes 1 and 2, maintaining comparable precision and recall to both LC1 and LC2. Notably, LC3 displays high recall (80%) for Mode 3. This performance could be attributed to using LSTM layers in the AE. This shows the modes identified by configuration 3 that utilize LSTMs are more suitable for rides that guarantee straight trajectories, such as city roads. Since straight rides make up most of the trajectories in city roads, the LSTM layers used in the AE benefitted from the longer temporal occurrences of class 3. When it comes to Mode 2, both LC3 and LC1 show higher recall and precision compared to LC2,

Class (Mode)	Precision	Recall	F1-Score	Support
1	0.55	0.70	0.61	16976
2	0.56	0.50	0.53	5975
3	0.57	0.72	0.64	14242
4	0.34	0.01	0.02	8368
Accuracy			0.55	45561
Macro Avg	0.50	0.48	0.45	45561
Weighted AVg	0.52	0.55	0.50	45561

Table 8.2: Classification report for LSTM classifier using Modes from DAE BN2 with an IPL of 1.0s

Class (Mode)	Precision	Recall	F1-Score	Support
1	0.58	0.50	0.53	10934
2	0.46	0.42	0.44	2165
3	0.69	0.92	0.72	25137
4	0.30	0.02	0.05	7440
Accuracy			0.65	45676
Macro Avg	0.51	0.47	0.45	45676
Weighted AVg	0.59	0.65	0.59	45676

Table 8.3: Classification report for LSTM classifier using Modes from DAE BN2 with an IPL of 1.6s

but it should be noted that even though LC2 has the lowest f1-score, it encapsulates trajectories that are mostly associated with negative roll angles (Roll Angle IQR within -15 deg to -30 deg) and roll rates (Roll rate IQR within 0.1 rad/s to -30 rad/s). These negative values are pertinent with left turn events, showcasing that LC2 is capable of predicting modes that are very safely critical since most accidents are associated with left turn events. All LSTM classifier models face consistent difficulties in predicting instances within Mode 4, showcasing low precision and recall, indicating inherent challenges in classification for this mode. This could be due to the fact that Mode 4 is derived from all three classes and lacks distinct characteristics such as Mode 1, Mode 2, and Mode 3. The differential performance across modes suggests considering specific model strengths and weaknesses when emphasizing different modes for predictive accuracy. Future research might explore strategies to improve predictions specifically for Mode 4, aiming to enhance overall classification accuracy across these distinct modes.

Class (Mode)	Precision	Recall	F1-Score	Support
1	0.62	0.70	0.66	14942
2	0.50	0.51	0.50	3556
3	0.64	0.80	0.71	19543
4	0.30	0.02	0.05	7556
Accuracy			0.62	45597
Macro Avg	0.52	0.51	0.48	45597
Weighted AVg	0.57	0.62	0.57	45597

Table 8.4: Classification report for LSTM classifier using Modes from LSTMAE BN2 with an IPL of 1.6s

8.2.1 Integrating a Threshold Condition

Based on the observations from the classification reports, since the precision and recall metrics are notably low for class 4 across all LSTM classifiers, a threshold of 0.5 is implemented to determine the classifier's output. This specific value is chosen since the weighted average accuracy observed across all three classifiers is greater than 0.49. This threshold-based approach ensures a stringent criterion for the classifier's output. A classifier without the threshold would select the modal LSTM with the highest probability, but with this threshold condition in place, we force the classifier to select LSTM4 when the highest probability across all the modes is less than 0.5. This is to make sure that the classifier doesn't end up choosing Mode 2 and Mode 1, which are associated with higher lateral dynamics, which could be fatal if the prediction is wrong. Hence, it is safer to select Mode 4 in low probability cases since Mode 4 has been trained on diverse scenarios (Mix of classes 1, 2, and 3 in the future 1.6s to 2.2s).

The labels, obtained through the unsupervised representation learning approach, are directly employed for MTP without further predictions or adjustments. Although an alternative class label classification method achieved approximately 90% accuracy using a specialized LSTM classifier for label prediction, concerns regarding potential model complexity and the introduction of errors steer the decision to use directly acquired unsupervised labels in this setup. Moreover, employing these directly acquired unsupervised labels serves a dual purpose. Firstly, it allows for assessing the MTP model with the LSTM classifiers, avoiding potential inaccuracies from class label predictions. Secondly, it provides insights into MTP's functionality when utilizing labels directly derived from the LS, yaw rate, and pitch rate combination.

The evaluation criterion chosen for assessing final roll angle predictions is the overall RMSE. This evaluation focuses on the mode predicted as most probable by the classifier, aligning the assessment with specific mode classifications. To ensure precision in evaluation, the RMSE calculation consistently references the mode predicted as most probable by the classifier, aiming to provide a reliable assessment of the hybrid model's performance under an LSTM classifiers that have weighted average accuracy less than 60%.

Model	Overall RMSE in Ideal case (deg)	Overall RMSE with M4 set to LSTM4 (deg)	Overall RMSE with M4 set to BL (deg)
MTP with LC1	6.850	8.3755	8.1697
MTP with LC2	6.917	8.2290	8.17319
MTP with LC3	6.9247	8.2903	8.0206

Table 8.5: Comparison of overall roll angle predictions across all Models

8.3 MTP with LSTM Classifiers

The evaluation of the model based on the current accuracy of classifiers revealed a decrease in roll angle prediction performance compared to the ideal scenario. This decline can be predominantly attributed to the lower accuracies observed in the LSTM classifiers. These findings emphasize the potential for enhancing the model's predictive capabilities, indicating that improving the accuracy of these LSTM classifiers could substantially bolster overall performance. A compelling illustration of this observation was the substitution of LSTM4 with mode 4, resulting in a slight enhancement in roll angle predictions (See Fig.8.6 and .8.7). This replacement proved effective because LSTM4 had been trained on samples where no consistent mode existed within the middle 0.8s window of the prediction horizon. Consequently, LSTM4, trained on scenarios with varying modes, exhibited behavior more akin to the baseline model. This adjustment underscored the influence of training data characteristics on model performance and its alignment with specific prediction scenarios. The summary of these results is shown in Tab. 8.6.

A comparison between the overall RMSE of roll angle predictions from all MTP models and their corresponding ideal scenarios highlighted the critical impact of accurate future mode information on predictive performance. Examining 8.8, it becomes apparent that in the ideal case, the classifiers' roll angle predictions exhibit an improvement starting around 1.5 seconds into the future. Notably, the window from 2.0 to 3.5 seconds in the prediction horizon showcases a substantial enhancement in roll angle predictions. This improvement coincides with the modes calculated within the middle 0.8 seconds, specifically from 1.6 to 2.2 seconds.

Conversely, the consequences of inaccurate future information are evident in the overall RMSE of LSTM classifiers with their actual accuracy. Roll angle predictions deteriorate notably within the same 2.0 to 3.0 seconds window, where the future mode information is crucial. Given the suboptimal performance of the LSTM classifiers, the MTP model with poorly constructed classifiers is prone to yielding inadequate roll angle predictions. This emphasizes the pivotal role of accurate future mode information in enhancing predictive accuracy and underscores the importance of well-performing classifiers within the MTP framework.

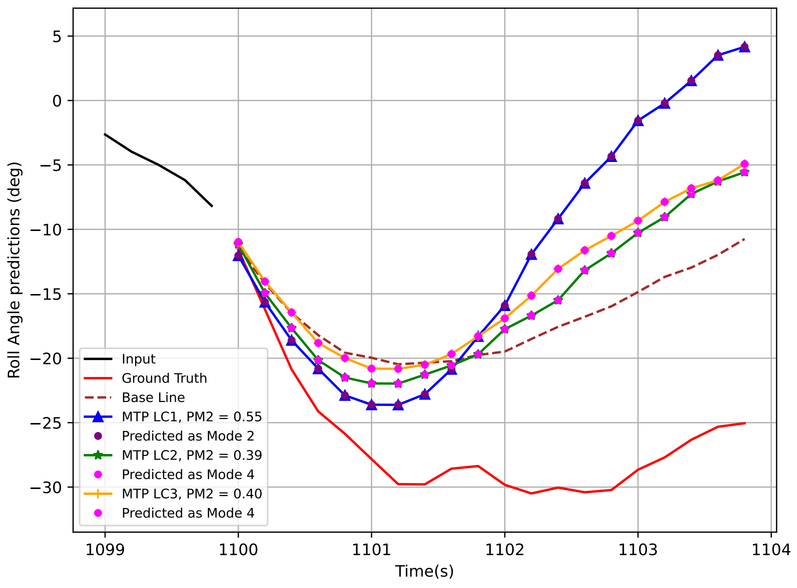


Figure 8.6: Comparison of Roll Angle Predictions across all Models.

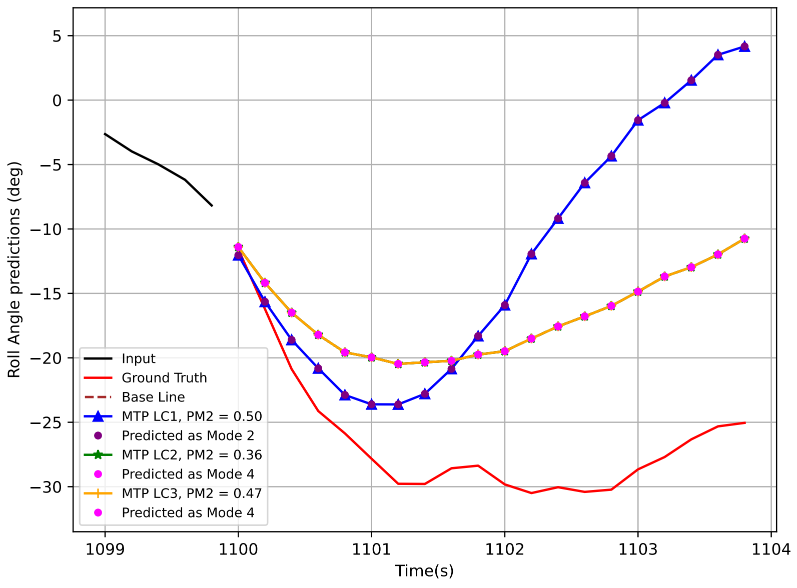


Figure 8.7: Comparison of Mode 2 Predictions across all Models after Switching Modes 4 of MTP(with LSTM Classifier) with Base Line Model

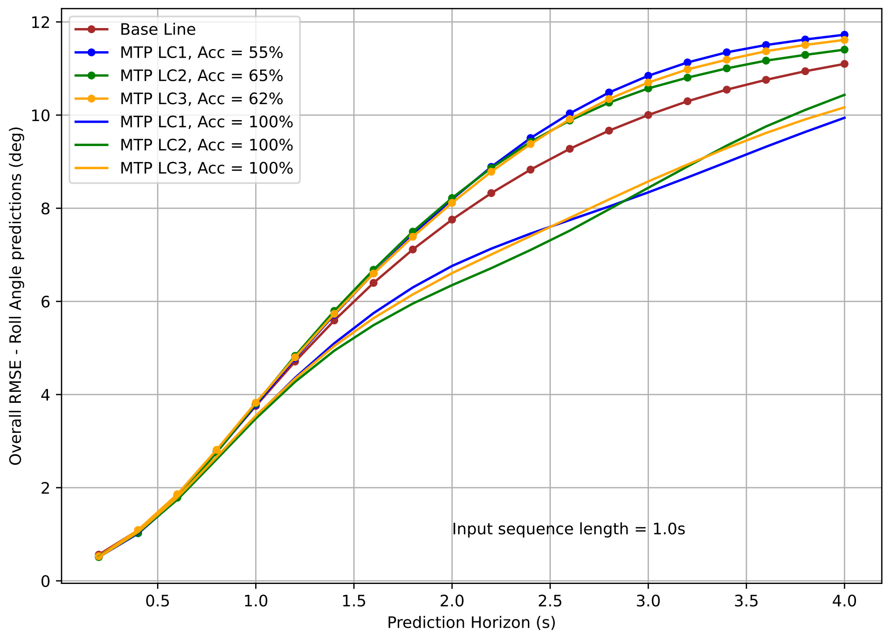


Figure 8.8: Overall RMSE at each point in Prediction Horizon

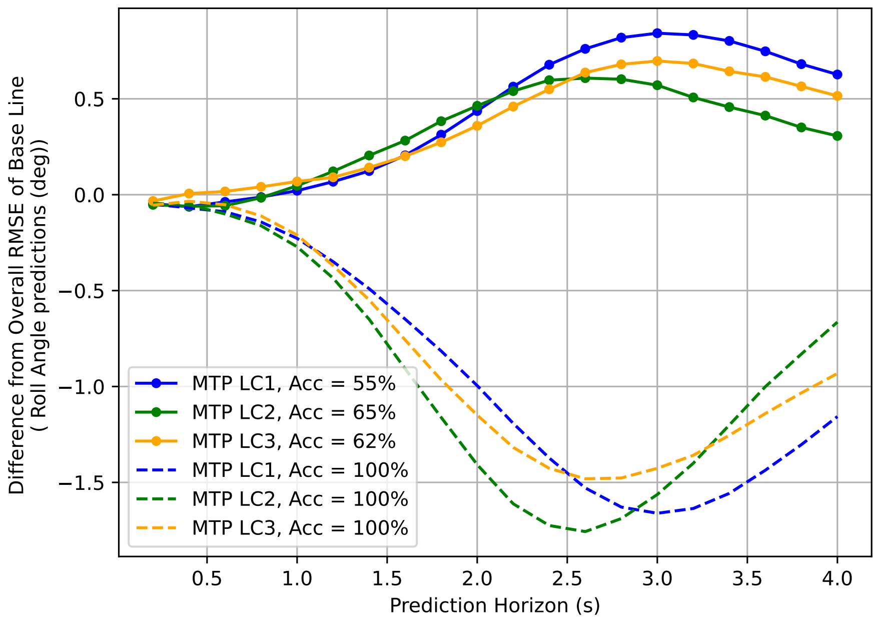


Figure 8.9: Difference in overall RMSE of Roll Angle values across the Prediction Horizon.

Model	Improvement over BL in Ideal case (% deg)	Performance gap in actual case (LSTM4 set to M4) (% deg)	Performance gap in actual case (BL set to M4) (% deg)
MTP with LC1	−12.82	+6.62	+3.99
MTP with LC2	−11.95	+4.64	+4.04
MTP with LC3	−11.85	+5.53	+2.09

Table 8.6: Comparison of Overall Roll Angle Predictions across All Models

9 Conlusion and Outlook

In this study, our primary objective centered on establishing a data-driven framework for MTP, leveraging vehicle dynamics data and selected rider behavior signals exclusively. This strategic methodological approach aimed to significantly enhance roll angle predictions, thereby contributing to the progressive development of active safety systems for PTWs.

Our methodology revolved around applying DL techniques, employing AE models to condense complex vehicle data, and implementing GMM for unsupervised clustering. Notably, our method effectively discerned high vehicle dynamic activities, such as turning and lane changes, from activities with intermediate and low dynamics, such as straight riding. Furthermore, the data segregation logic helped us realize consistent modes across the prediction horizon, enabling us to use just one LSTM for each mode conveniently. These observations provided a positive answer to research questions **RQ1** and **RQ2**. This accomplishment emphasizes the achievement of implementing MTP using solely the original dataset. It eliminates the necessity for a dataset rich in GPS information and the context of the surrounding vehicle offers another affirmative response to **RQ3**. Under ideal conditions, our model exhibited notable success, showcasing a remarkable 12.8% reduction in the overall RMSE of the roll angle predictions. However, challenges surfaced in predicting realistic scenarios, highlighting the necessity for further refinement, particularly within the LSTM classifiers. The contrast in our model's performance under ideal and real-world conditions highlighted the importance of precise future mode information to enhance predictive accuracy. It also demonstrated the potential of MTP. This illumination emphasized the significant role of robust and accurate classifiers within our data-driven MTP framework. Therefore, to comprehensively address **RQ4** in a more realistic setting, further research is required to enhance the accuracy of LSTM classifiers.

The classifiers' performance faced considerable challenges primarily due to the class imbalance within the dataset's identified modes. Although various data augmentation frameworks exist to rectify this imbalance, their implementation poses intricate challenges. A major concern was the potential introduction of mislabeling issues, primarily since our labeling process was generated via an unsupervised method. Mitigating these mislabeling risks while utilizing data augmentation became a complex task within the scope of our study. Additionally, while VAE methods offer promise in data augmentation and class balancing through a generative approach, their exploration within our study was constrained by the intricate nature of validating the authenticity of the generated data. Specifically, in our case, the vehicle dynamics data for PTW needs to adhere to unique physics governing PTW stability. Therefore, due to the complexity and potential risks associated with mislabeling, coupled with the intricate validation required for the generative approach, the exploration of VAE for data augmentation and class balancing can be addressed in future studies.

In conclusion, this study's core achievement not only contributes significantly to the potential of enhancing roll angle predictions but also lays a foundation for future advancements in active safety systems, thereby fostering safer journeys for PTW riders.

Bibliography

[Abd20] B. Abdelmoudjib. *Artificial Intelligence for Vehicle Behavior Anticipation: Hybrid Approach Based on Maneuver Classification and Traject...* Apr. 1, 2020. URL: https://ouci.dntb.gov.ua/en/works/40K1EEo7/ (visited on 12/03/2023) (cit. on p. 28).

[AHK01] C. C. Aggarwal, A. Hinneburg, D. A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *Database Theory — ICDT 2001*. Ed. by J. Van den Bussche, V. Vianu. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 420–434. ISBN: 978-3-540-44503-6. DOI: 10.1007/3-540-44503-X_27 (cit. on p. 49).

[AL18] F. Altché, A. de La Fortelle. *An LSTM Network for Highway Trajectory Prediction*. Jan. 24, 2018. DOI: 10.48550/arXiv.1801.07962. arXiv: 1801.07962[cs]. URL: http://arxiv.org/abs/1801.07962 (visited on 12/03/2023) (cit. on pp. 22, 27).

[BMT+21] L. Brown, A. Morris, P. Thomas, K. Ekambaram, D. Margaritis, R. Davidse, D. S. Usami, M. Robibaro, L. Persia, I. Buttler, A. Ziakopoulos, A. Theofilatos, G. Yannis, A. Martin, F. Wadji. "Investigation of accidents involving powered two wheelers and bicycles – A European in-depth study". In: *Journal of Safety Research* 76 (Feb. 1, 2021), pp. 135–145. ISSN: 0022-4375. DOI: 10.1016/j.jsr.2020.12.015. URL: https://www.sciencedirect.com/science/article/pii/S0022437520301651 (visited on 12/03/2023) (cit. on p. 17).

[BVG18] E. N. Barmpounakis, E. I. Vlahogianni, J. C. Golias. "Identifying Predictable Patterns in the Unconventional Overtaking Decisions of PTW for Cooperative ITS". In: *IEEE Transactions on Intelligent Vehicles* 3.1 (Mar. 2018). Conference Name: IEEE Transactions on Intelligent Vehicles, pp. 102–111. ISSN: 2379-8904. DOI: 10.1109/TIV.2017.2788195. URL: https://ieeexplore.ieee.org/document/8241853 (visited on 12/03/2023) (cit. on p. 29).

[CKPW20] Z. Cui, R. Ke, Z. Pu, Y. Wang. *Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Forecasting Network-wide Traffic State with Missing Values*. May 23, 2020. DOI: 10.48550/arXiv.2005.11627. arXiv: 2005.11627[cs, eess,stat]. URL: http://arxiv.org/abs/2005.11627 (visited on 12/03/2023) (cit. on pp. 23, 24).

[CMC19] CMC. *Time will Explain - Sequence Processing with Recurrent Networks*. Oct. 2019. URL: https://web.stanford.edu/~jurafsky/slp3/old_oct19/9.pdf (visited on 12/03/2023) (cit. on p. 22).

[Con] C. M. Consortium. *Path Prediction for PTWs*. https://www.car-2-car.org/documents/general-documents. Accessed December 4, 2023 (cit. on p. 29).

[DT18] N. Deo, M. M. Trivedi. “Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018 IEEE Intelligent Vehicles Symposium (IV). ISSN: 1931-0587. June 2018, pp. 1179–1184. DOI: 10.1109/IVS.2018.8500493. URL: https://ieeexplore.ieee.org/document/8500493 (visited on 12/03/2023) (cit. on p. 64).

[DYY+19] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, R. Salakhutdinov. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. June 2, 2019. DOI: 10.48550/arXiv.1901.02860. arXiv: 1901.02860[cs, stat]. URL: http://arxiv.org/abs/1901.02860 (visited on 12/03/2023) (cit. on p. 27).

[GHCG21] F. Giuliari, I. Hasan, M. Cristani, F. Galasso. “Transformer Networks for Trajectory Forecasting”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2020 25th International Conference on Pattern Recognition (ICPR). ISSN: 1051-4651. Jan. 2021, pp. 10335–10342. DOI: 10.1109/ICPR48806.2021.9412190. URL: https://ieeexplore.ieee.org/abstract/document/9412190 (visited on 12/03/2023) (cit. on p. 27).

[GSC+21] H. Gao, H. Su, Y. Cai, R. Wu, Z. Hao, Y. Xu, W. Wu, J. Wang, Z. Li, Z. Kan. “Trajectory prediction of cyclist based on dynamic Bayesian network and long short-term memory model at unsignalized intersections”. In: *Science China Information Sciences* 64.7 (May 18, 2021), p. 172207. ISSN: 1869-1919. DOI: 10.1007/s11432-020-3071-8. URL: https://doi.org/10.1007/s11432-020-3071-8 (visited on 12/03/2023) (cit. on p. 28).

[HAK] A. Hinneburg, C. C. Aggarwal, D. A. Keim. “What is the nearest neighbor in high dimensional spaces?” In: () (cit. on p. 33).

[HBCY13] A. Houenou, P. Bonnifait, V. Cherfaoui, W. Yao. “Vehicle trajectory prediction based on motion model and maneuver recognition”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. ISSN: 2153-0866. Nov. 2013, pp. 4363–4369. DOI: 10.1109/IROS.2013.6696982. URL: https://ieeexplore.ieee.org/abstract/document/6696982 (visited on 12/03/2023) (cit. on pp. 28, 34).

[HMD+20] X. Huang, S. G. McGill, J. A. DeCastro, L. Fletcher, J. J. Leonard, B. C. Williams, G. Rosman. *DiversityGAN: Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling*. Mar. 21, 2020. DOI: 10.48550/arXiv.1911.12736. arXiv: 1911.12736[cs]. URL: http://arxiv.org/abs/1911.12736 (visited on 12/03/2023) (cit. on p. 29).

[HSW+22] D. Hutchins, I. Schlag, Y. Wu, E. Dyer, B. Neyshabur. *Block-Recurrent Transformers*. Nov. 1, 2022. DOI: 10.48550/arXiv.2203.07852. arXiv: 2203.07852[cs]. URL: http://arxiv.org/abs/2203.07852 (visited on 12/03/2023) (cit. on p. 27).

[HWP+21] Z. Huang, J. Wang, L. Pi, X. Song, L. Yang. “LSTM based trajectory prediction model for cyclist utilizing multiple interactions with environment”. In: *Pattern Recognition* 112 (Apr. 1, 2021), p. 107800. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107800. URL: https://www.sciencedirect.com/science/article/pii/S0031320320306038 (visited on 12/03/2023) (cit. on p. 30).

[JHAC23] F. Jalti, B. Hajji, A. Acri, M. Calì. "An Advanced Rider-Cornering-Assistance System for PTW Vehicles Developed Using ML KNN Method". In: *Sensors* 23.3 (Jan. 2023). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 1540. ISSN: 1424-8220. DOI: 10.3390/s23031540. URL: https://www.mdpi.com/1424-8220/23/3/1540 (visited on 12/03/2023) (cit. on p. 29).

[LDL+19] J. Li, B. Dai, X. Li, X. Xu, D. Liu. "A Dynamic Bayesian Network for Vehicle Maneuver Prediction in Highway Driving Scenarios: Framework and Verification". In: *Electronics* 8.1 (Jan. 1, 2019), p. 40. ISSN: 2079-9292. DOI: 10.3390/electronics8010040. URL: https://www.mdpi.com/2079-9292/8/1/40 (visited on 12/03/2023) (cit. on p. 28).

[LHH+18] X. Lyu, M. Hueser, S. L. Hyland, G. Zerveas, G. Raetsch. *Improving Clinical Predictions through Unsupervised Time Series Representation Learning*. Dec. 2, 2018. DOI: 10.48550/arXiv.1812.00490. arXiv: 1812.00490[cs,stat]. URL: http://arxiv.org/abs/1812.00490 (visited on 12/03/2023) (cit. on p. 29).

[LL21] S. Lu, R. Li. *DAC: Deep Autoencoder-based Clustering, a General Deep Learning Framework of Representation Learning*. Feb. 15, 2021. DOI: 10.48550/arXiv.2102.07472. arXiv: 2102.07472[cs]. URL: http://arxiv.org/abs/2102.07472 (visited on 12/03/2023) (cit. on p. 29).

[MHZ+21] B. Mersch, T. Höllen, K. Zhao, C. Stachniss, R. Roscher. *Maneuver-based Trajectory Prediction for Self-driving Cars Using Spatio-temporal Convolutional Networks*. Sept. 15, 2021. DOI: 10.48550/arXiv.2109.07365. arXiv: 2109.07365[cs]. URL: http://arxiv.org/abs/2109.07365 (visited on 12/03/2023) (cit. on p. 64).

[MMCS11] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. Ed. by T. Honkela, W. Duch, M. Girolami, S. Kaski. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 52–59. ISBN: 978-3-642-21735-7. DOI: 10.1007/978-3-642-21735-7_7 (cit. on p. 20).

[MSKD23] S. Mozaffari, M. A. Sormoli, K. Koufos, M. Dianati. *mozaffariMultimodalManoeuvreTrajectory2023 for Automated Driving on Highways Using Transformer Networks*. July 26, 2023. DOI: 10.48550/arXiv.2303.16109. arXiv: 2303.16109[cs]. URL: http://arxiv.org/abs/2303.16109 (visited on 12/03/2023) (cit. on p. 29).

[Mur12] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012 (cit. on pp. 52, 76).

[OMCG17] O. Olabiyi, E. Martinson, V. Chintalapudi, R. Guo. *Driver Action Prediction Using Deep (Bidirectional) Recurrent Neural Network*. June 7, 2017. DOI: 10.48550/arXiv.1706.02257. arXiv: 1706.02257[cs,stat]. URL: http://arxiv.org/abs/1706.02257 (visited on 12/03/2023) (cit. on p. 27).

[Ord] M. Ordonez. *Registrations in key European markets in the first half of 2023 confirm growing interest for L-category vehicles*. URL: https://acem.eu/registrations-in-key-european-markets-in-the-first-half-of-2023-confirm-growing-interest-for-l-category-vehicles (visited on 12/03/2023) (cit. on p. 17).

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on pp. 22, 23).

[PKG21] E. A. I. Pool, J. F. P. Kooij, D. M. Gavrila. "Crafted vs Learned Representations in Predictive Models—A Case Study on Cyclist Path Prediction". In: *IEEE Transactions on Intelligent Vehicles* 6.4 (Dec. 2021). Conference Name: IEEE Transactions on Intelligent Vehicles, pp. 747–759. ISSN: 2379-8904. DOI: 10.1109/TIV.2021.3064253. URL: https://ieeexplore.ieee.org/document/9372805 (visited on 12/03/2023) (cit. on p. 30).

[PMLL20] X. Peng, Y. L. Murphey, R. Liu, Y. Li. "Driving maneuver early detection via sequence learning from vehicle signals and video images". In: *Pattern Recognition* 103 (July 1, 2020). ADS Bibcode: 2020PatRe.10307276P, p. 107276. DOI: 10.1016/j.patcog.2020.107276. URL: https://ui.adsabs.harvard.edu/abs/2020PatRe.10307276P (visited on 12/03/2023) (cit. on p. 28).

[PSCH22] G. Pang, C. Shen, L. Cao, A. v. d. Hengel. "Deep Learning for Anomaly Detection: A Review". In: *ACM Computing Surveys* 54.2 (Mar. 31, 2022), pp. 1–38. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3439950. arXiv: 2007.02500[cs,stat]. URL: http://arxiv.org/abs/2007.02500 (visited on 12/04/2023) (cit. on p. 21).

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 51).

[PWW+23] Z. Pan, Y. Wang, K. Wang, H. Chen, C. Yang, W. Gui. "Imputation of Missing Values in Time Series Using an Adaptive-Learned Median-Filled Deep Autoencoder". In: *IEEE Transactions on Cybernetics* 53.2 (Feb. 2023). Conference Name: IEEE Transactions on Cybernetics, pp. 695–706. ISSN: 2168-2275. DOI: 10.1109/TCYB.2022.3167995. URL: https://ieeexplore.ieee.org/document/9768200 (visited on 12/03/2023) (cit. on pp. 23, 25).

[QFP+21] A. Quintanar, D. Fernández-Llorca, I. Parra, R. Izquierdo, M. A. Sotelo. *Predicting Vehicles Trajectories in Urban Scenarios with Transformer Networks and Augmented Information*. June 7, 2021. DOI: 10.48550/arXiv.2106.00559. arXiv: 2106.00559[cs]. URL: http://arxiv.org/abs/2106.00559 (visited on 12/03/2023) (cit. on p. 27).

[Rey09] D. Reynolds. "Gaussian Mixture Models". In: *Encyclopedia of Biometrics*. Ed. by S. Z. Li, A. Jain. Boston, MA: Springer US, 2009, pp. 659–663. ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_196. URL: https://doi.org/10.1007/978-0-387-73003-5_196 (visited on 12/03/2023) (cit. on p. 51).

[SM19] R. C. Staudemeyer, E. R. Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. Sept. 12, 2019. DOI: 10.48550/arXiv.1909.09586. arXiv: 1909.09586[cs]. URL: http://arxiv.org/abs/1909.09586 (visited on 12/03/2023) (cit. on p. 22).

[Ste00] M. Stephens. “Dealing With Label Switching in Mixture Models”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 62.4 (Nov. 1, 2000), pp. 795–809. ISSN: 1369-7412, 1467-9868. DOI: 10.1111/1467-9868.00265. URL: https://academic.oup.com/jrsssb/article/62/4/795/7083253 (visited on 12/03/2023) (cit. on p. 53).

[SWS23] K. L. Stolle, A. Wahl, S. Schmidt. “Trajectory Forecasting for Powered Two Wheelers by Roll Angle Prediction with an LSTM Network”. In: The Evolving Scholar - BMD 2023, 5th Edition. Sept. 7, 2023. DOI: 10.59490/64e61a33563addeb42473c8f. URL: https://dapp.orvium.io/deposits/64e61a33563addeb42473c8f/view (visited on 12/03/2023) (cit. on pp. 19, 80).

[TS19] Y. C. Tang, R. Salakhutdinov. *Multiple Futures Prediction*. Dec. 6, 2019. DOI: 10.48550/arXiv.1911.00997. arXiv: 1911.00997[cs,stat]. URL: http://arxiv.org/abs/1911.00997 (visited on 12/03/2023) (cit. on p. 29).

[Uni23] T. P. S. University. *2.5 - The Coefficient of Determination, r-squared | STAT 462*. 2023. URL: https://online.stat.psu.edu/stat462/node/95/ (visited on 12/03/2023) (cit. on p. 40).

[Węg18] S. Węglarczyk. “Kernel density estimation and its application”. In: *ITM Web of Conferences* 23 (2018). Ed. by W. Zielinski, L. Kuchar, A. Michalski, B. Kazmierczak, p. 00037. ISSN: 2271-2097. DOI: 10.1051/itmconf/20182300037. URL: https://www.itm-conferences.org/10.1051/itmconf/20182300037 (visited on 12/03/2023) (cit. on p. 46).

[ZJP+21] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, C. Eickhoff. “A transformer-based framework for multivariate time series representation learning”. In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2114–2124 (cit. on p. 28).

[ZRK+22] S. Zernetsch, H. Reichert, V. Kress, K. Doll, B. Sick. “A Holistic View on Probabilistic Trajectory Forecasting – Case Study. Cyclist Intention Detection”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022 IEEE Intelligent Vehicles Symposium (IV). June 2022, pp. 265–272. DOI: 10.1109/IV51971.2022.9827220. URL: https://ieeexplore.ieee.org/document/9827220 (visited on 12/03/2023) (cit. on p. 30).

[ZZCH18] J. Zhai, S. Zhang, J. Chen, Q. He. “Autoencoder and Its Various Variants”. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC). ISSN: 2577-1655. Oct. 2018, pp. 415–419. DOI: 10.1109/SMC.2018.00080. URL: https://ieeexplore.ieee.org/document/8616075 (visited on 12/03/2023) (cit. on p. 20).

[ZZTL22] Y. Zhang, Y. Zou, J. Tang, J. Liang. “A Lane-Changing Prediction Method Based on Temporal Convolution Network”. In: *Transportmetrica B: Transport Dynamics* 10.1 (Dec. 31, 2022), pp. 849–863. ISSN: 2168-0566, 2168-0582. DOI: 10.1080/21680566.2021.1950072. arXiv: 2011.01224[cs,eess]. URL: http://arxiv.org/abs/2011.01224 (visited on 12/03/2023) (cit. on p. 28).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature