UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Elisabet Hein

Automating the Classification of Disengagements Using Convolutional Neural Networks

Bachelor's Thesis (9 ECTS)

Supervisor:
Dietmar Alfred Paul Kurt Pfahl, PhD

Tartu 2025

Automating the Classification of Disengagements Using Convolutional Neural Networks

Abstract:

This thesis presents a fully automated solution for classifying disengagement events in autonomous driving using Convolutional Neural Networks. The goal is to replace the current manual classification process with a more efficient and scalable approach. The solution includes eight binary classification models and corresponding data preprocessing scripts, enabling automated categorization. Future improvements could focus on refining data preprocessing and evaluating model performance on larger datasets.

Keywords: Autonomous Driving, Automated Classification, Disengagement Event Analysis, Convolutional Neural Networks, Explainable Artificial Intelligence

CERCS: P170 Computer Science, numerical analysis, systems, control

Juhtimise ülevõtmiste klassifitseerimise automatiseerimine kasutades konvolutisoonilisi närvivõrke

Lühikokkuvõte:

Selles lõputöös esitatakse täielikult automatiseeritud lahendus autonoomse sõidu ülevõtmiste klassifitseerimiseks, kasutades konvolutsioonilisi närvivõrke. Eesmärgiks on asendada praegu kasutusel olev manuaalne ülevõtmiste klassifitseerimine tõhusama ja paremini skaleeritava meetodiga. Tulemusteks on kaheksa binaarset klassifitseerivat mudelit ning andmete eeltöötlemiseks loodud skriptid. Võimalikud tulevased edasiarendused hõlmavad andmetöötlusprotsessi täiustamist ning mudelite treenimist ja testimist suurematel andmekogumitel.

Võtmesõnad: Autonoomne sõitmine, Automaatne klassifitseerimine, Ülevõtmiste analüüs, Konvolutsioonilised närvivõrgud, Seletatav tehisintellekt

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Table of Contents

1. Introduction

Autonomous driving technology has been advancing rapidly during the past decade. Chu et al. [1], Khan et al. [2], and Karla et al. [3] have described how it is expected to play a significant role in humans' daily transportation. These authors emphasize its potential to improve road safety by reducing traffic-related fatalities and to enhance overall driving comfort.

The term "safety driver" has come into use in the context of automated driving system (ADS) testing and usage [1]. Safety driver is a human supervisor who is required to sit behind the vehicle's steering wheel and maintain awareness of the ADS and take control over driving when necessary.

Chu et. al. [1] and McGehee et al. [4] also emphasize that entirely automated vehicles have not yet been fully implemented nor tested, due to safety, technical or ethical reasons. To ensure the reliability of those vehicles, continuous improvements and testing are critical. Therefore, as of now, a safety driver is needed to supervise ADS and maintain situational awareness at all times in most automated vehicles.

The transition process from autonomous driving to manual driving is called a **disengagement event** [5] - an instance where a safety driver takes control over driving and proceeds to continue driving manually. Events like these can be analyzed to find potential bugs in the software or possible improvement points, for example in perception or decision-making algorithms, as disengagements could be triggered by various reasons.

Disengagements are typically classified into two main categories:

- **Planned disengagements** - Situations where the safety driver will intentionally take over driving or where the system is programmed to hand over control. This could be due to legal or predefined safety reasons, such as approaching a pedestrian crossing.
- **Unplanned disengagements** - Situations where the safety driver unexpectedly needs to take over driving. The reasons could include problems in the system, safety risks or the safety driver's personal evaluation on the situation.

Planned and unplanned disengagements can be further sub-classified based on specific reasons for the events. Classifying those events can help analyze the system's behavior and refine the safety and performance of the software.

The Autonomous Driving Lab (ADL) [6] of the University of Tartu plays a crucial role in evaluating the current state of autonomous driving, performing research in this data-driven field and building a software for an autonomous vehicle. The ADL also focuses on improving autonomous vehicle technology by collecting system data from test drives and performing post-drive analysis. Currently, disengagement events are reviewed by examining image and video data to determine the reasons and causes behind those events. This process has been done manually, which therefore takes a lot of human resources to go through all the data. Excel sheets have been used to analyze the data and keep track of different rides and the classifications, which could lead to human errors and takes up a significant amount of time. This process could be automated.

A semi-automated solution has been previously proposed by Pirjo Jõelo in 2024 with her Bachelor's thesis "Automating Classification of Disengagements using Foxglove" [7]. The study was conducted in the ADL, with the aim to present a semi-automated Disengagement Classification Method (DCM) for classifying disengagements via a rule-based approach. The proposed solution used FoxGlove[1] tool with a custom panel in the application, to classify disengagements as either planned or unplanned, and additionally sub-classify the planned events based on the reason into five known sub-categories. However, this solution still required manual inspection, and did not focus on sub-classifying both main categories.

The goal of this thesis project is to propose a solution for fully automated disengagements classification by using Convolutional Neural Networks (CNNs). The thesis aims to leverage deep learning algorithms to classify all possible disengagement events into their respective categories, or where classification is uncertain, leave those unlabeled. By fully automating this process, it will be possible to reduce human workload, improve the efficiency and possible mistakes, and ensure the consistency of classification results.

The goals of the thesis project are:

1. **Develop a structured data processing pipeline** to transform raw ADL-provided data into a format suitable for neural network training.
2. **Design and implement an automated classification system** for disengagement events using binary convolutional neural network models, each specialized for a specific disengagement class.

[1] A vizualisation tool for robotics developers. https://foxglove.dev/ (24.04.2025)

3. **Evaluate model performance** by testing the trained models on disengagement events from each class, ensuring reliable classification.
4. **Utilize explanatory methods** to interpret model decisions, providing insights into the classification process and overall system behavior.

The thesis is structured as follows:

- **Background**: This section describes the concept of Automated Vehicles, Autonomous Driving Lab and its work and their used technologies in depth, provides a description of disengagement events and their classes. Describes convolutional neural networks and other possible approaches that were considered.
- **Methodology**: This section describes the steps taken to gather the data, preprocess data into a suitable format for neural network models, extracting disengagement events from the whole data, selecting appropriate model structures, training the models, validating the results, and using explanatory methods to interpret models' decision-making process.
- **Results**: This section provides the results of the thesis and tests. It also includes explanatory methods used to interpret the models' work and decisions.
- **Discussion**: This section covers the analysis of the findings of the proposed approach, comparing them with the existing semi-automated method, listing potential improvements and discussing future research directions.
- **Appendices**: The appendices provide a comprehensive analysis of the results from the explainable AI methods.

To ensure clarity and coherence, ChatGPT[2] was utilized in all sections of the thesis to refine the formatting and structure of the text. No new content was generated; rather, the existing text was reworded and reorganized for improved readability and flow.

[2] OpenAI (2025). ChatGPT (version GPT-4o mini): https://chatgpt.com/ (24.02.2025).

2. Background

The following section provides a comprehensive overview of the key concepts relevant to this thesis. It is divided into several subsections, beginning with an explanation of autonomous driving levels, detailing the varying degrees of system automation and human supervision. This is followed by an overview of the Autonomous Driving Lab, including its research focus and technical aspects. Next, a detailed description of disengagement events is presented, along with the classification structure used by the ADL. Finally, to support the proposed automation of disengagement classification, different classification models are discussed, outlining potential approaches and the rationale behind the final chosen method.

2.1 Levels of Automation in Automated Driving Systems

Autonomous vehicles have been a major focus of research and development in recent years, with the potential to revolutionize transportation by improving safety, efficiency, and accessibility [1, 2, 3]. By leveraging advanced sensors, artificial intelligence, and machine learning, these vehicles can assist or even replace human drivers in various driving tasks. However, the journey to fully autonomous driving is complex, requiring continuous advancements in technology, regulation, and infrastructure.

To better understand where this technology currently stands, SAE International (previously known as the Society of Automotive Engineers) [8] has defined six levels of driving automation, ranging from Level 0 (no automation) to Level 5 (fully automated). These levels help classify vehicle capabilities and the degree of human involvement. The visual representation is shown on Figure 1. The levels are as follows:

- Level 0: No Driving Automation;
- Level 1: Driver Assistance;
- Level 2: Partial Driving Automation;
- Level 3: Conditional Driving Automation;
- Level 4: High Driving Automation;
- Level 5: Full Driving Automation.

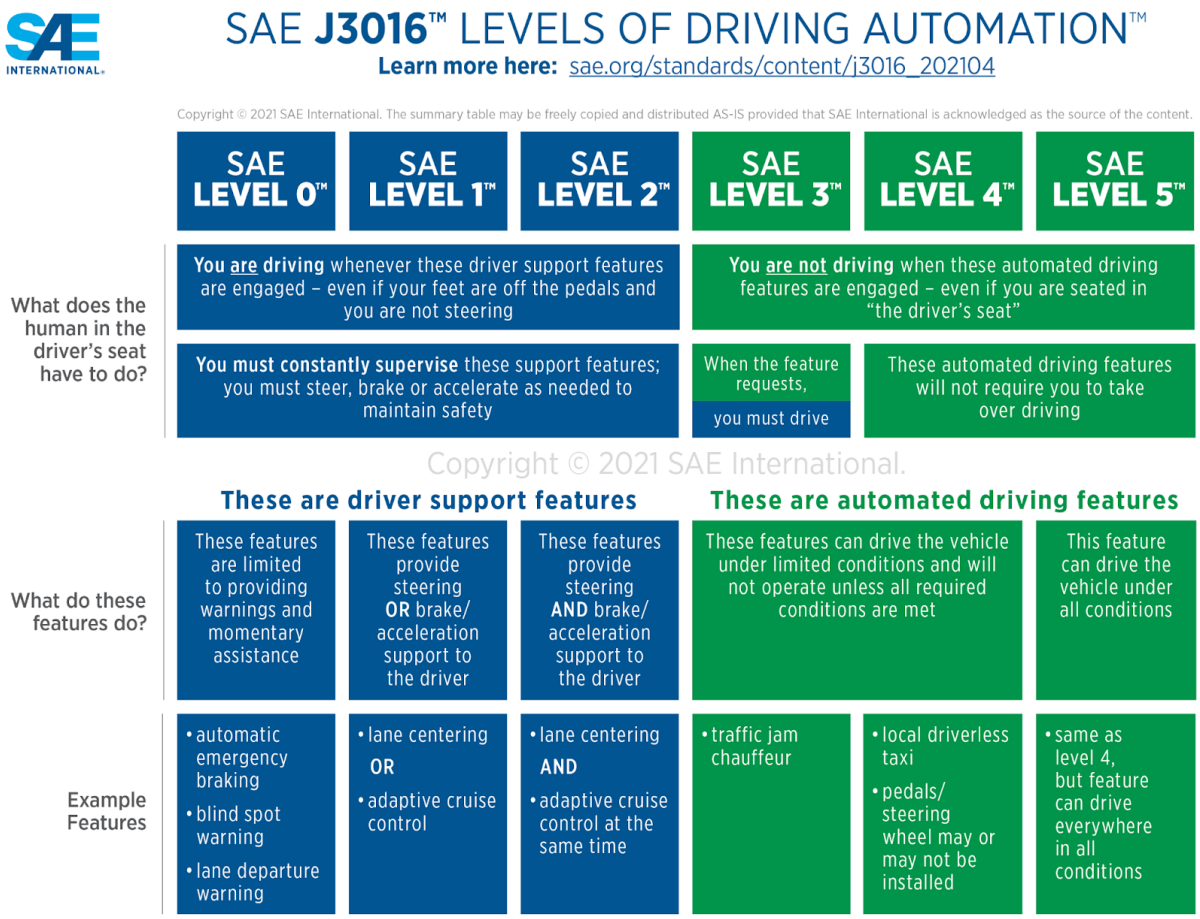

Figure 1. SAE International definition of "Levels of Driving Automation" [8].

Modern vehicles primarily operate at **Level 0**. This level does not automate driving but can provide warnings or simple assistance during driving, like automatic emergency braking or blind spot warnings.

Level 1 and **Level 2** are often referred to as Advanced Driver-Assistance Systems (ADAS), which provide partial automation. With these levels, the driver is still responsible for driving but is assisted by the driving system, like steering, braking, and accelerating.

At **Level 3**, vehicles would be able to handle certain driving tasks autonomously, but only under specific conditions. The human driver is still required to stay attentive and take over driving when necessary.

Level 4 automated vehicles can perform driving on their own without human intervention, in predefined scenarios, like highways or urban areas. Yet, a human driver must still take control if necessary.

Level 5, the highest level of automation, will perform all driving tasks under all conditions and no intervention of the driver is needed. The vehicle can operate fully independently.

The vehicle used in this study falls into the category of partial automation, operating at a level between SAE Level 2 and Level 3 - referred to as Level 2.5 [6]. This means that while the vehicle can perform certain automated driving tasks, the driver remains responsible for overall control and must be prepared to intervene when necessary.

Chu et al. [1] also highlight in their research about autonomous vehicle safety, how for various reasons, like technical, legislative, or ethical reasons, fully automated vehicles have not been widely implemented or tested yet. A human supervisor, safety driver, will be therefore needed in autonomous vehicles for a long period until fully automated driving systems are implemented, validated and tested to an acceptable extent.

2.2 The Autonomous Driving Lab

The Autonomous Driving Lab (ADL) [6] of the University of Tartu is a research laboratory, which was founded in 2019, in cooperation with the Estonian mobility unicorn[3] Bolt. ADL aims to evaluate the technology readiness of self-driving software and does research in autonomy software, high-definition maps, behavior prediction, motion planning, teleoperation, learned driving, validation and testing, perception uncertainty, human-vehicle interaction and security.

ADL uses a Lexus RX 450h SUV (Figure 2), which is equipped with all sensors necessary, like lidars, cameras, a radar, and Global Navigation Satellite System (GNSS) equipment, to perform test drives with the vehicle running on in-house software [6].

[3] A startup company with value over one billion USD.

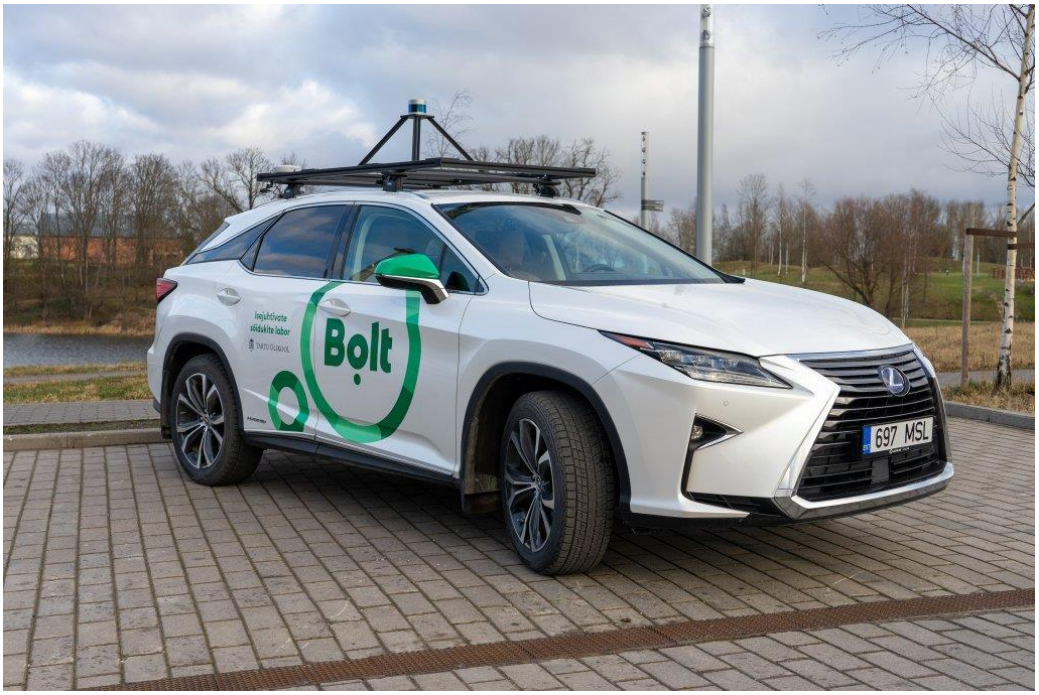

Figure 2. The Lexus RX 450h SUV used by the ADL for test drives [6].

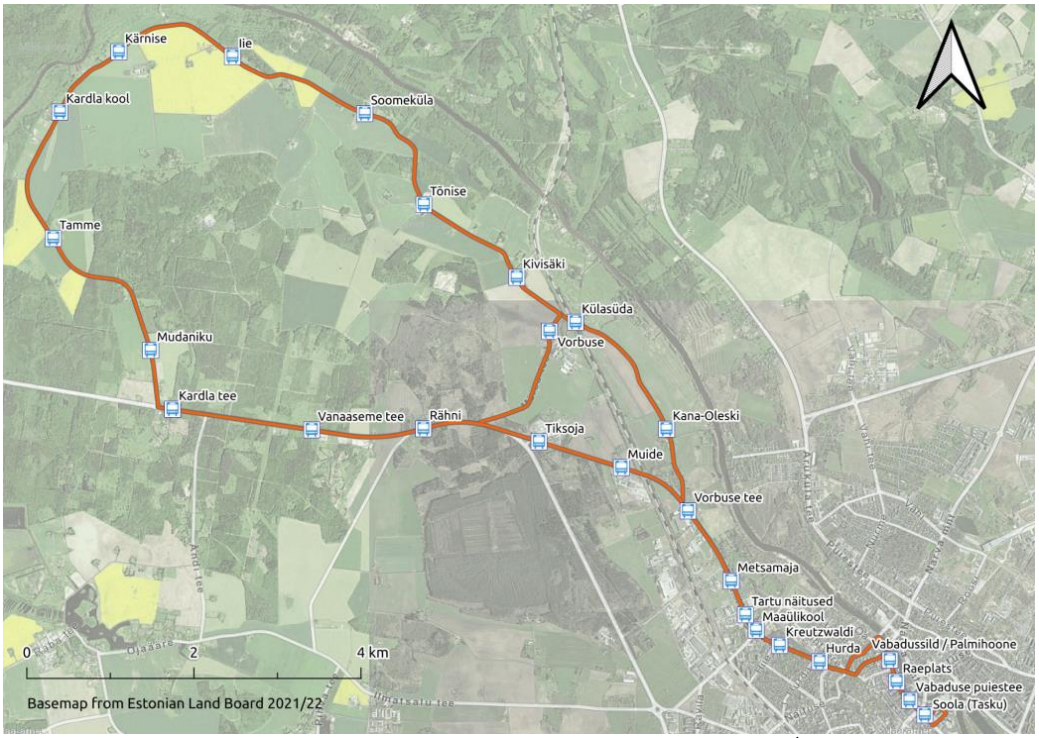

Figure 3. Full route of the autonomous car.[4]

The test drives used in this thesis were conducted from the 16th of October 2023 to 3rd of November 2023. Each selected drive was performed on the route visible on Figure 3. The technologies used for the software and data analysis will be described in the next sub-section.

[4] Figure 3 is based on a personal communication with ADL reasearch engineer Edgar Sepp.

2.3 Autonomous Driving Lab Technologies

Autonomous Driving Lab [6] uses a Python[5]-based software called Autoware Mini, which is using the ROS 1 platform.

Robot Operating System (ROS) [9] is a platform for software developers that provides different tools and libraries, which can be used in robot-based systems. For example, ROS will provide tools for implementing and operating parts of the robot system, and handling communication between different system components. ROS is an open-source system, which is running only on Unix[6]-based operating systems. Bag file format is used by the ROS for saving logs and messages from different components of the system. The platform will log messages from nodes[7] to topics[8]. A self-driving car system has, for example, components that are responsible for the operation of gas and brake pedals. Messages from those components could be logged into one topic, which describes pedal data.

ADL performs post-drive analysis using ROS bags, as well as analyzing the disengagement events, which is the main focus of this thesis.

2.4 Disengagement Events

A disengagement event in the autonomous vehicle's driving process, is an instance where the safety driver sitting behind the steering wheel is required or decides to take control over driving. This is a transition from autonomous driving to manual driving. Disengagements can be classified as planned and unplanned like described in the Introduction section. In the ADL further sub-classification is done based on the reason for the disengagements - unplanned disengagements divided into 4 and planned disengagements divided into 5 sub-categories.

Unplanned disengagements' sub-categories:

1. **"OBS"** - Situations where an obstacle is blocking the vehicle.
2. **"Safety"** - Situations where the safety driver decides to take over diving for their safety based on their own judgement, for example unstable driving.
3. **"Bad engage"** - Situations where the vehicle was previously disengaged and engaged again, but the problem remained, therefore resulting in a disengagement again.

[5] A programming language. https://www.python.org/ (24.04.2025)
[6] Unix is an operating system, developed in the 1960s.
[7] A communication channel for different parts of the system to send and receive messages.
[8] A process or a component of the system that performs small tasks like controlling sensors.

4. **“Localization”** - Situations where the vehicle had problems with localizations, like misinterpreting its location or not getting enough signals from satellites.

Planned disengagements’ sub-categories:

1. **“Pedestrian crossing”** - Situations where a pedestrian was intending to cross the cross-walk, which requires the safety driver to disengage for the safety of the pedestrian.
2. **“Give way”** - Situations where the vehicle is approaching a “give way” intersection and the safety driver is required to take over in order to prevent any collisions.
3. **“Turnback”** - Situations where the vehicle needs to perform a turnback, for example turn back to drive back the same road it originally came from. This is done in manual mode because the autonomous vehicle in use cannot drive backwards autonomously.
4. **“SPEED”** - Situations where the maximum allowed speed on the road exceeded the legally allowed one for the autonomous vehicle (50 km/h), therefore manual driving is required to be used.
5. **“STOP”** - Situations where the vehicle has stopped in a bus stop and needs to either do a turnback or drive back to the main road.

This thesis uses these sub-categories and propose an automated solution for classifying disengagements into the listed sub-classes.

2.5 Classification Models

The goal of this study is to develop an effective machine learning model to classify disengagement events from log data. The model needs to accurately distinguish different disengagement categories while minimizing misclassifications, given the safety-critical nature of the application.

To achieve this, multiple machine learning approaches were explored in a pre-study, including anomaly detection using autoencoders, multi-class classification, and convolutional neural networks (CNNs). Each approach had its advantages and limitations, ultimately leading to the selection of a binary classification framework using CNNs as the most effective method.

2.5.1 Anomaly Detection Using Autoencoders

One of the initial approaches explored, was the use of autoencoders for anomaly detection in the log data. Autoencoder is a type of neural network used in unsupervised learning, designed to reconstruct input data by learning its underlying patterns [10]. It consists of an encoder,

which compresses the input to a lower-dimensional representation, and a decoder, which tries to reconstruct the original data from the compressed form.

The hypothesis was, when encountering an anomaly - like a disengagement event - the autoencoder would flag such cases as deviations from normal driving data. This approach aimed to facilitate real-time detection of potentially unsafe situations or bugs in the software, with a possibility to incorporate additional classification layers for further analysis of anomalies. However, this approach resulted in a high false positive rate, making it unreliable for a safety-critical task. Since minimizing misclassifications was a key requirement, this method was deemed unsuitable, leading to a shift toward supervised learning techniques.

2.5.2 Multi-Class Classification Model

Another approach explored was a multi-class classification model, where a single model was trained to differentiate between all disengagement categories. The motivation was to train a single unified classification framework rather than a separate model for each class. However, the performance of the multi-class models was suboptimal, with low accuracy and poor generalization to unseen data. The complexity of distinguishing multiple disengagement types within a single model, combined with limited training data, likely contributed to these limitations.

2.5.3 Binary Classification Using CNNs

Given the challenges faced with previously tried methods, the study shifted towards a binary classification approach using Convolutional Neural Networks (CNNs). Traditionally, CNNs are most commonly associated with image processing. However, they can also be highly effective for time-series data, particularly through 1D Convolutional Layers (Conv1D), which can be useful in tasks involving one-dimensional sequence data, such as audio analysis, time-series forecasting, or natural language processing [11]. Conv1D layers aim to extract meaningful features that contribute to the model's task at hand which would require understanding patterns in the data based on time and order.

Petnehazi [12] describes in their research how CNNs utilize sliding local receptive fields, enabling them to detect local patterns and correlations in sequential data. This is particularly useful for time series, where nearby values often hold meaningful dependencies.

Petnehazi [12] also describes how Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are often considered an optimal choice for sequential data. However, they also describe how recent studies have shown

that convolutional neural networks could work even better in situations with a sequence modeling task. Their ability to capture short- to mid-range dependencies makes them suitable for tasks where disengagement events are characterized by distinct, localized patterns in vehicle control and sensor data. And, given that disengagement events involve sudden changes in normal driving behavior, CNNs promised to be well-suited to identify those abrupt transitions in the log data, making them a strong choice for automated classification.

3. Methodology

The methodology chapter outlines the process of developing disengagement classification models, from data preparation and model training to evaluation and interpretability. This includes key phases, steps, and tasks involved in building and validating the models, as well as applying explainability techniques to understand their decision-making. A summarized overview of this methodology is presented in Figure 4, which visually represents the structured approach taken in this study.

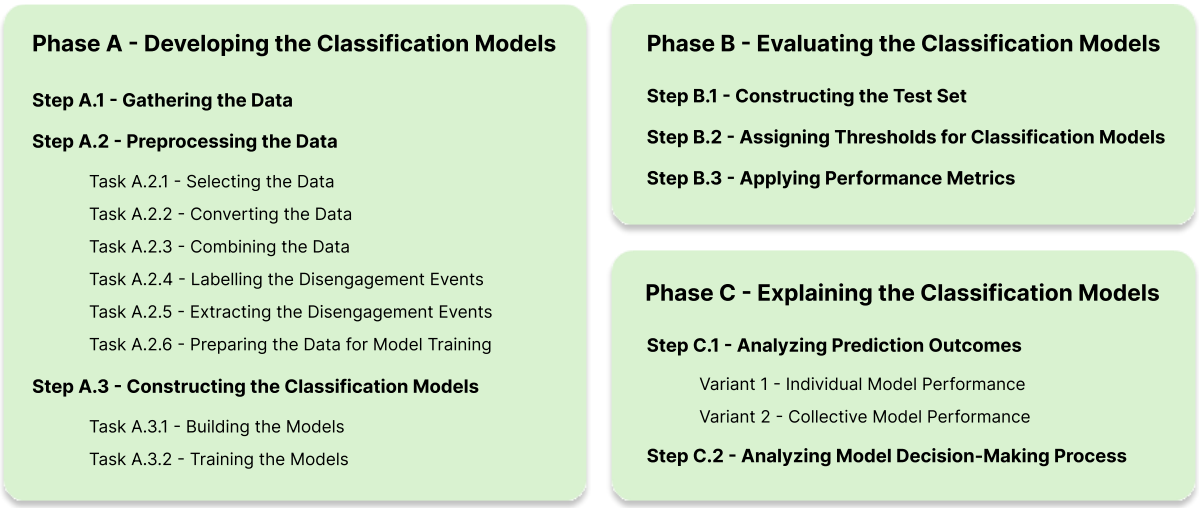

Figure 4. Overview of the methodology phases, steps, and tasks.

3.1 Phase A: Developing the Classification Models

This section explains the development of disengagement classification models, covering dataset selection, preprocessing techniques, and model architecture.

3.1.1 Step A.1: Gathering the Data

To develop classification models for categorizing disengagements, a high-quality dataset must be obtained from a reliable source, such as a laboratory or company database. The dataset is selected based on a specific timeframe or, if applicable, sourced from previous research to ensure compatibility and enable comparative analysis.

The dataset must contain multiple instances of disengagements for each class being modeled. This guarantees that each class has sufficient samples for both training and testing.

Since autonomous driving vehicles typically log data in ROS (.bag) format, which can be large and storage-intensive, it is advisable to manage data storage efficiently. External solid-state

drives (SSDs) can be used to accommodate large datasets if local hardware resources are limited.

3.1.2 Step A.2: Preprocessing the Data

This section covers steps taken to clean, convert, and structure the raw data to make it suitable for machine learning, including selecting relevant variables, handling missing values, and standardizing formats.

Task A.2.1: Selecting the Data

Log data from autonomous vehicles often include information from numerous components responsible for vehicle operation. Not all logged topics are relevant to disengagement classification. For example, data streams intended solely for visualization (e.g., camera overlays) or peripheral system states (e.g., traffic light signals) typically do not contribute meaningfully to the classification of disengagement events and can therefore be excluded from processing. The objective is to identify and retain only essential data, such as:

- vehicle control commands;
- position and velocity data;
- localization information;
- object detection outputs;
- trajectory planning details.

Excessive data selection may strain computational resources, such as random access memory (RAM). Therefore, it is recommended to start with a minimal set of crucial topics and expand only if necessary.

Task A.2.2: Converting the Data

To facilitate further preprocessing, collected ROS .bag files are converted into a standardized format, such as comma-separated values (CSV). Since ROS .bag files are optimized for Unix-based operating systems, but this research is conducted on a Windows machine, CSV is chosen for its widespread use in machine learning applications.

The conversion process could be automated, allowing a script or tool to accept a ROS .bag file as input and generate separate CSV files for each selected topic. This streamlines the conversion process, reduces manual effort, and maintains modularity for future analysis.

Task A.2.3: Combining the Data

Once the relevant topics are converted to CSV format, additional preprocessing is required to refine the dataset:

- Remove irrelevant columns such as component IDs and redundant values.
- Merge topic files into a single, unified CSV per ride using timestamps as the primary key, and keeping all unique timestamps.

As ROS logs data asynchronously, timestamps across different topics may not always align, leading to missing values. To solve this problem, Pandas[9] library's functions forward filling and backward filling can be used.

Forward fill is a function which fills all missing or invalid values, like NaN (Not a Number) values, with the last valid observed value [13]. This function can be used to ensure that each column's values remain the same until the next valid value is logged.

Backward fill is a function which fills all missing or invalid values with the next observed valid value [13]. This function can be used to ensure that all columns are filled even for those components that start their logging process later than the first timestamp. With this approach all missing values that cannot be filled with forward filling are replaced with a valid value.

With these techniques, missing values are replaced with valid values, maintaining the consistency of the dataset while also preserving the relationships across all topics. The result will be a single combined CSV file per each ride, containing all selected topics and columns.

Task A.2.4: Labelling the Disengagement Events

Since the goal is to classify disengagement events, it is essential to enrich the log files with relevant disengagement information. To achieve this, it is recommended to use a script that identifies transitions from autonomous mode to manual mode - typically detected through a column indicating the drive mode in the log data. These transitions are labeled as disengagements in a new column, allowing for precise identification of the timestamps where disengagement events occurred.

Additionally, since the classification task focuses on disengagement events, an additional column specifying the type of disengagement must be included. Each disengagement type is

[9] Open source data analysis and manipulation tool in Python. https://pandas.pydata.org/ (27.01.2025)

recommended to be mapped to numerical values rather than text labels for consistency and ease of processing. A script can automate this process by assigning numerical class labels to timestamps where disengagements are detected, while non-disengagement timestamps can be assigned a placeholder value (e.g., -1).

Task A.2.5: Extracting the Disengagement Events

To effectively classify disengagement events, these instances will be isolated from the complete ride dataset. This process involves selecting a predefined timeframe that captures data leading up to each disengagement event. A recommended extraction window ranges from 3 to 10 seconds; however, the optimal length may vary depending on the dataset characteristics and computational constraints. To determine the most suitable duration, the following considerations can be taken into account:

- **The timeframe contains only one disengagement event**. If multiple events fall within the selected timeframe, it is advisable to shorten the window accordingly.
- **Sufficient data** must be available for disengagement events occurring early in the ride. If an event takes place within the first *x* seconds of the ride, the timeframe may need to be shortened, or the disengagement event may be excluded from training and testing.
- **Computational resources** must be considered. A longer timeframe increases memory usage, as all disengagement events are processed simultaneously during training. Therefore, it is essential to ensure that the system has adequate RAM capacity to handle the selected timeframe length.

Each disengagement event will be stored as a separate CSV file, containing only the extracted timeframe.

Task A.2.6: Preparing the Data for Model Training

To effectively train the models, a standardized structure for the input dataframe must be established. This involves the following steps:

1. **Test Set Extraction**
   Reserve a portion of the original dataset as the test set before any data augmentation. This ensures the test set remains entirely untouched, providing a truly independent and representative subset for evaluating model performance. Ensure this set includes at least

one sample from each class, preserving class diversity and supporting meaningful evaluation metrics.

2. **Data Augmentation**

   Ideally, each class has at least a few hundred samples to ensure sufficient training data. If the dataset is imbalanced or lacks sufficient samples per class, generate additional data to maintain a fixed number of samples per class.

   The optimal number of samples can be determined through model performance testing, but keep in mind:

   a. Smaller datasets should have fewer generated samples to prevent overfitting.
   b. Larger datasets can accommodate more generated samples without compromising generalization.

   Augmentation can be performed using methods like data jittering, which introduces slight variations by adding noise to the original data.

3. **Converting Categorical and Textual Data to Numerical Representations**

   Since neural networks require numerical inputs, all string-based values must be transformed. This can be achieved using one-hot encoding or categorical mapping (assigning numerical values to text-based categories).

   In this thesis, categorical mapping will be used to convert string values into numerical ones.

4. **Normalizing Numerical Values**

   Scale all numerical features to a standard range, typically 0 to 1, using techniques like min-max scaling, standard scaling, or robust scaling.

5. **Defining a Fixed Structure for Disengagement Events**

   Set a fixed length for all disengagement event dataframes by either padding or truncating rows as necessary. Since disengagement files generally contain a similar number of rows due to the predefined time window (e.g., a fixed x-second timeframe), determine the maximum number of rows across all disengagements and set that as the standard length. This ensures that no data is lost, and shorter disengagements are padded with minimal additional rows to maintain consistency.

6. **Splitting Data for Training and Validation**

   For each target class (for each binary classification model to be trained):

   a. **Select an equal number of samples from other classes**. Ensure that the number of randomly selected samples from each of the other classes is the same to maintain balance across all classes.

   b. **Combine the target class with the selected negative class samples** to create a balanced dataset for each binary classification model.

   c. **Split the data into training and validation sets with an 80/20 split**. Ensure that both the training and validation sets have an equal number of positive (target class) and negative (other classes) samples. This guarantees balanced class distribution in both sets.

      - **Training Set (80%)** - Used for model learning.
      - **Validation Set (20%)** - Used for fine-tuning during training.

7. **Defining Feature Sets (X and y)**

   Once the data is split into training and validation sets for each target class and its corresponding binary model, define the feature sets for both sets:

   - **X (Independent Variables):** A combination of target-class windows and an equal number of randomly selected windows from other classes.
   - **y (Dependent Variable):** Binary labels, where 1 represents the target class and 0 represents all other classes.

By following this structured preprocessing approach, the dataset remains balanced, standardized, and optimized, providing a strong foundation for robust model training.

3.1.3 Step A.3: Constructing the Classification Models

This section describes the process of training the disengagement classification models, including architecture selection, hyperparameter tuning, and optimization techniques.

Task A.3.1: Building the Models

For each disengagement class, a separate binary classification model will be trained. Each model is designed to determine whether a given input window belongs to the specific class it was trained to identify.

The input to the model is a disengagement event sequence with dimensions (timestamps, features), where:

- Timestamps represent the number of time points within a single timeframe.
- Features correspond to the number of columns in the dataframe.

The output is a probability score indicating the likelihood that the input belongs to the target class.

The chosen architecture must be well-suited for time-series data, with the ability to extract meaningful features and relationships. For this research **1D Convolutional Neural Network (Conv1D) layers** are used, which are effective for sequential data by applying convolutional filters across time-series patterns.

To enhance feature extraction and model performance, additional layers can be incorporated:

- **Kernel Regularization** - Helps prevent overfitting and enhancing the model's generalization ability [14].
- **Batch Normalization** - Normalizes intermediate activations for more stable and faster training [15].
- **MaxPooling1D** - Downsamples the sequence, retaining the most significant patterns and most important features [16].

As this is a binary classification task, the output layer consists of a single neuron with a sigmoid activation function, which converts predictions into probability scores between 0 and 1.

For this research **Adam (Adaptive Moment Estimation) optimizer** with a customized learning rate of 0.0001 is used, ensuring a gradual and stable convergence, suitable for this binary classification task with potentially complex features. Harjai et. al. [17] describe in their research how Adam is a popular deep learning optimization technique, as it has a rather simple configuration, and is suitable for a majority of problems. Kingma et. al. [18] also describe in their research about Adam how it is computationally efficient, requiring little memory and suits data with large numbers of rows or features, like in this thesis.

For the loss function, **binary cross-entropy** is used, as it is specifically designed for binary classification problems. Kulkarni et al. [19] identify binary cross-entropy as the foremost choice when it comes to binary classification. This loss function ensures that both false negatives (where the true label is 1 but the model predicts 0) and false positives (where the true label is 0 but the model predicts 1) are given equal importance, thereby maintaining a balanced evaluation of classification errors.

To prevent overfitting and unnecessary computations, **EarlyStopping** is implemented as a callback. This stops training once a monitored metric (e.g., validation loss) ceases to improve, ensuring the model halts at an optimal point.

Additionally, if the dataset is imbalanced, **class weights** are applied to ensure the model gives equal attention to both classes.

Task A.3.2: Training the Models

Each disengagement class will have its own model, all trained using the same CNN architecture to ensure consistency across models.

The number of training epochs is determined through preliminary test runs. If the validation loss continues to decrease, increasing the number of epochs is recommended to allow the model sufficient training time for optimal performance. However, if EarlyStopping is used, the chosen epoch count is recommended to align with the point where EarlyStopping is triggered - indicating that further training would lead to overfitting and is unnecessary.

To allow for frequent weight updating, a small value for the batch size is used. This helps capture fine-grained patterns in the data. Experimenting with batch sizes of 8, 16, or 32 is recommended, and selecting the most effective size based on the model's performance.

3.2 Phase B: Evaluating the Classification Models

This section outlines the methods used to assess model performance, including test set generation, classification threshold adjustments, and key performance metrics.

3.2.1 Step B.1: Constructing the Test Set

As outlined in the data preparation section (Task A.2.6), each class should have at least one sample reserved for testing. These test samples must be entirely excluded from the training process to ensure an accurate evaluation on unseen data.

If the original dataset contains a sufficient number of samples per class (e.g., several dozen), extracting multiple test samples from each class is recommended to create a more realistic and diverse test set.

If the dataset lacks enough test samples, the same data augmentation techniques used for training will be applied to generate additional test samples. To ensure a reliable evaluation, it is recommended to have **at least five test samples per disengagement class**. Five samples per

class ensures a balance between reliability and redundancy - too few may lead to biased evaluation results, while too many offer limited additional value, as augmented samples are derived from the same original data and therefore tend to have similarities.

3.2.2 Step B.2: Assigning Thresholds for Classification Models

Each model applies a default classification threshold of 0.5, meaning that if a model predicts a sample's label with a probability of 0.5 or higher, the sample is assigned that label.

The classification threshold can be adjusted based on model performance. For example, if a model produces too many false positives, raising the threshold may help by only accepting predictions with higher confidence. This situation may indicate that test samples from different classes share similar patterns, making classification more challenging.

3.2.3 Step B.3: Applying Performance Metrics

To assess model performance, the following standard classification metrics are used for each binary model separately, based on the test set and the correctness of its predictions:

- **Accuracy**: Measures the proportion of correctly classified instances (Equation 1).
- **Precision**: The ratio of correctly predicted positive observations to total predicted positives (Equation 2).
- **Recall**: The ratio of correctly predicted positive observations to all actual positives (Equation 3).
- **F1-score**: The harmonic mean of precision and recall, balancing both metrics (Equation 4).

Each metric is computed individually for every binary model, using the test set and distinguishing between correct and incorrect predictions. The formulas used for calculation are:

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{true negatives} + \text{false negatives} + \text{false positives}} \quad (1)$$

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (2)$$

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (3)$$

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

This ensures that each class-specific model is evaluated independently, reflecting its performance in distinguishing between positive and negative cases.

3.3 Phase C: Explaining the Classification Models

Interpreting machine learning models is crucial, especially in safety critical fields, like autonomous driving and its post-drive analysis, such as the disengagement classification. This phase focuses on analyzing the prediction outcomes of the models and applying SHapley Additive exPlanations (SHAP) to understand their decision-making process.

3.3.1 Step C.1: Analyzing Prediction Outcomes

To evaluate model performance, prediction outcomes are analyzed at both the individual model level and across the entire set of models. This is done by examining the distribution of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The analysis consists of two key approaches:

Variant 1: Individual Model Performance

By analyzing prediction outcomes (TP, TN, FP, FN) for each model, it is possible to identify well-performing models and those prone to frequent misclassifications. This analysis helps uncover potential issues, such as:

- **Class Similarity:** Some classes may share overlapping features, leading to frequent misclassification.
- **Lack of Distinguishing Features:** Certain classes may lack clear characteristics, making them difficult for models to classify correctly.
- **Threshold Adjustment Needs:** The classification threshold may need refinement to improve accuracy.
- **Data Limitations:** Additional training data may be required to help models better distinguish between similar patterns.

Variant 2: Collective Model Performance

Since multiple binary models contribute to the final classification, a decision process is required to determine the most appropriate label for each sample. The final label assignment follows these rules:

- **Single Prediction**: For a given sample, if exactly one model predicts its class with a probability ≥ 0.5, then the predicted class becomes the final label of the sample.
- **Conflicting Predictions**: For a given sample, if multiple models predict their respective classes with probabilities ≥ 0.5, then the final label assigned to the sample is "Uncertain".
- **No Confident Prediction**: For a given sample, if none of the models predict their class with a probability ≥ 0.5, then the final label assigned to the sample is "Undetermined".

Ideally, each sample receives a single, correct prediction. However, achieving this requires all models to perform consistently well. The following are common misclassification scenarios:

- A weak model fails to identify its target class (FN), while another model produces an incorrect positive prediction (FP), leading to an inaccurate final label.
- A combination of false positives and a true positive across models, leading to conflicting predictions.
- A single weak model introducing errors that could lead to no confident predictions or influence the overall classification outcome.

To further understand these interactions, predictions are categorized based on their surrounding predictions ("context") within the set of models (Table 1):

Table 1. Correctness of predictions and their dependence on other models for the same sample, represented in a format of "good" and "bad" context.

Correctness	"Good" Context (Reliable Context)	"Bad" Context (Unreliable Context)
TP	All other models predict TN	At least one FP exists
TN	One TP, all others TN	Multiple FPs and/or no TP
FP	One TP, all others TN	Multiple FP and/or no TP
FN	All other models predict TN	At least one FP exists

Table 1 highlights that the final label is highly dependent on all models performing well. A single underperforming model can introduce "bad company" for otherwise correct predictions, leading to more uncertain or incorrect final labels. This analysis can guide decisions on whether to exclude weak models or retrain them to improve overall reliability.

3.3.2 Step C.2: Analyzing Model Decision-Making Process

Deep learning models can achieve high accuracy and performance metrics, yet their decision-making process can often remain a “black box”. Therefore, Explainable Artificial Intelligence (XAI), such as SHapley Additive exPlanations (SHAP), or Local Interpretable Model-Agnostic Explanations (LIME), methods will be applied to interpret the model’s predictions.

For this study, **SHAP** is selected as the interpretability technique. Jhumka et al. [20] have described in their research where they used SHAP, how it provides numeric values that quantify the contribution of each feature to a model’s predictions, offering insights into which factors were most influential in classification decisions. The goal of using SHAP is to gain a deeper understanding of why the trained models make certain predictions and to identify which features are most important in distinguishing disengagement events. Cunha et al. [21] emphasize how in a binary classification setting, SHAP values indicate whether a feature positively, negatively, or neutrally influences the likelihood of a given class.

Using the SHAP values, various visualizations can be created to show the results in a visually appealing and understandable way, for example, using heatmaps or diagrams. By doing that, differences between the created models can be distinguished and explanations to the models’ decision-making can be found.

4. Results

This section presents the outcomes of the developed binary classification models for disengagement classification. The section is divided into three phases as presented in the Methodology section. It covers the key aspects of model development, including data preprocessing, training, and evaluation. The scripts used for data preparation, model training, and validation are discussed, along with performance metrics to assess model effectiveness. Additionally, explainability techniques, such as SHAP analysis, are explored to provide insights into feature importance and model decision-making.

4.1 Phase A: Set of Classification Models

A total of eight binary classification models were developed, each designed to predict a specific disengagement reason, and each using a typical convolutional neural network architecture. Each model independently determines whether a given input window corresponds to its respective disengagement reason class. Table 2 outlines the disengagement classes, with their respective numerical class, the name assigned to the model, and the file name of the model. The models can be found in a GitHub repository[10].

Table 2. The mapping between the disengagement classes and the numerical classes with the corresponding model name and model file name.

Disengagement Class	**Numerical Mapping**	**Model Name**	**Model File**
"OBS"	0	CL0	"model_class_0_final.h5"
"Safety"	1	CL1	"model_class_1_final.h5"
"Turnback"	2	CL2	"model_class_2_final.h5"
"Pedestrian crossing"	3	CL3	"model_class_3_final.h5"
"Localization"	4	CL4	"model_class_4_final.h5"
"Give way"	6	CL6	"model_class_6_final.h5"
"SPEED"	7	CL7	"model_class_7_final.h5"
"STOP"	8	CL8	"model_class_8_final.h5"

[10] URL: https://github.com/Elisabethein/Automating-the-Classification-of-Disengagements-using-Convolutional-Neural-Networks

There is no class 5 because the original "Bad Engage" label was removed and replaced with the most recent disengagement reason prior to the transition. This adjustment was made to ensure cleaner data for model training, as the "Bad Engage" label did not represent a distinct disengagement reason but rather an ongoing issue linked to a previously occurring class. By reassigning these instances to their preceding disengagement category, the dataset maintains a more precise and meaningful classification structure.

4.1.1 Model Architecture

All models follow a consistent CNN architecture (see Section 2.5.3. for the introduction to CNNs and their typical applications), employing a typical 1D convolutional neural network structure. The architecture remains identical across all models, with their key characteristics described below.

Input Shape

- **Input**: A sequence with dimensions (1304 timestamps, 484 features).

Feature Extraction Layers

- **Conv1D Layers**: Three convolutional layers with increasing filter sizes (128, 256, 512) were used to capture hierarchical features from the data.
- **Kernel Regularization** [14]: L2 regularization was applied to reduce overfitting.
- **Batch Normalization** [15]: Normalized the intermediate activations to stabilize and speed up training.
- **MaxPooling1D** [16]: Reduced the temporal resolution, focusing on the most prominent features while downsampling the sequence.

Global Feature Representation

- **GlobalAveragePooling1D** [16]: Aggregated temporal features into a single feature vector by computing the average of all values in the feature map.

Classification Head

- **Dense Layers**: Fully connected layers (with 128 units and a ReLU (rectified linear unit) activation) learning higher-level representations. Dropout (30%) was applied to prevent overfitting.

- **Output Layer**: A single neuron with a sigmoid activation function was used to output a probability score, indicating whether the input belonged to the target class.

The models were trained using the Adam optimizer (learning rate = 0.0001) with binary cross-entropy as the loss function. EarlyStopping was implemented, monitoring validation loss, halting training after five epochs of no improvement, and restoring the best weights. Class weights were computed to balance class importance.

4.1.2 Details of Classification Model Construction

All models were developed following the three-step process outlined in the Methodology section: data gathering, data preprocessing, and model construction. This section presents the detailed intermediate results corresponding to each of these steps and their corresponding sub-tasks.

To provide an overview of the dataset characteristics before delving into the detailed results, Figure 5 presents a high-level summary of the dataset and the preprocessing pipeline leading up to model training during Phase A. The pipeline begins with the raw data, which comprises 31 ROS bag files spanning 136 topics, each containing several thousand features.

During preprocessing, 27 ROS bag files were selected, narrowing the focus to 9 key topics which were selected based on relevance to the task. The feature space was reduced to 484 selected features. From these files, 160 disengagement events were extracted and segmented into fixed-length windows of 5 seconds, corresponding to 1304 timestamps per window. To provide further insight into the selected data, a diagram is included showing the feature distribution across the 9 topics. These topics are numbered consistently with the definitions given in the corresponding subsection for Task A.2.1.

Subsequently, the data was split into training and testing sets, resulting in 154 training samples and 6 test samples. Due to the class imbalance and the small number of samples, data augmentation techniques were applied to the training set to improve class balance and sample size. This process yielded a total of 2560 training samples and 30 test samples, with both sets achieving balanced class distributions.

This overview sets the stage for the following subsections, which detail the training set construction process. Section 4.2 will then focus specifically on the test set and its characteristics.

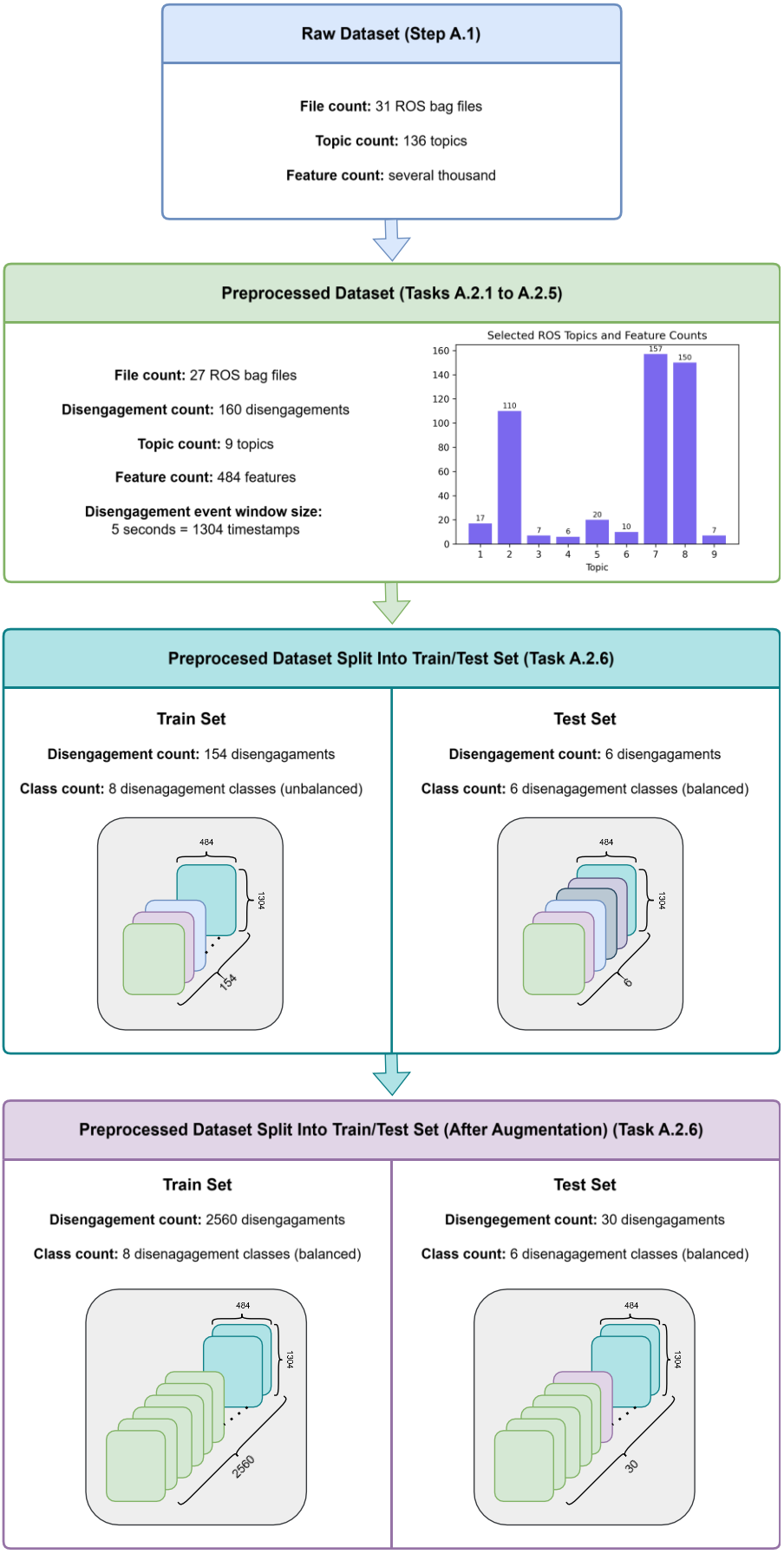

Figure 5. Characteristics of the data created and used in Phase A.

Step A.1: Data Gathering

A total of 27 ROS bag files, containing 160 disengagement events, were selected for this study, ensuring compatibility with the dataset used in Jõelo's thesis [7] and leveraging previously labeled disengagement events from the ADL's prior work.

The data was sourced from the University of Tartu's ROSBAG cloud server[11] and collected during test drives conducted between October 16 and November 3, 2023. Of the 31 available test rides, 4 were excluded due to corruption or processing issues, leaving 27 usable files. Each ROS bag file contained logs from various sections of the test trail, capturing data across 136 topics from different autonomous vehicle components. File sizes ranged from 20 GB to 70 GB, requiring SSD storage for efficient handling. Due to the size and format of the data, further preprocessing was necessary, which was performed in the next step.

The 27 files contained a total of 160 disengagement events, which are summarized in Table 3 with their respective counts. As mentioned in Section 4.1, the original "Bad Engage" label (class 5) was removed and replaced with the most recent disengagement reason prior to the transition to ensure cleaner data for model training.

Table 3. Disengagement classes with their respective counts.

Disengagement Class	**Numerical Mapping**	**Count**
"OBS"	0	65
"Safety"	1	30
"Turnback"	2	14
"Pedestrian crossing"	3	24
"Localization"	4	13
"Give way"	6	12
"SPEED"	7	1
"STOP"	8	1
SUM		**160**

[11] ROSBAG database. https://bagdatabase.cloud.ut.ee/ (27.10.2024) Direct access to the database has to be requested from the ADL.

Step A.2: Data Preprocessing

Step A.2 resulted in a preprocessed, augmented, and balanced dataset for model training and testing. The final dataset consisted of 2,566 five-second disengagement event dataframes, derived from 160 original dataframes. These were then split into training, validation, and test sets. A sample of one of the original preprocessed dataframes can be found in the following repository:

"*2023-10-16-10-14-49_tiksoja_ride_02_sfa_disengagement_1.csv*" (GitHub Respository).

This section follows the Step A.2 structure outlined in the Methodology and presents the results of each task, including data selection, conversion, merging, and final preparation for model training.

Selection of Relevant Topics (Task A.2.1)

The main outcome of Task A.2.1 was the selection of the most relevant topics for the study, done with the assistance of ADL research engineer Edgar Sepp. These topics were chosen based on their importance in capturing key aspects of vehicle control, localization, and environmental awareness:

1. **/control-vehicle_cmd -** High-level vehicle control commands, including acceleration, braking, steering, and gear changes.
2. **/detection-final_objects** - Data on all detected objects and obstacles identified by the vehicle's sensors.
3. **/localization-current_pose** - Information about the ego vehicle[12]'s position.
4. **/localization-current_velocity** - Velocity of the ego vehicle.
5. **/novatel-oem7-bestpos** - Most accurate GNSS-calculated position, including latitude, longitude, height, and GPS time offset.
6. **/novatel-oem7-inspva** - Data on the vehicle's position, velocity, and attitude (orientation).
7. **/planning-global_path** - Vehicle's planned global trajectory.
8. **/planning-local_path** - A subset of the global path used for local planning, object detection, and speed calculations.

[12] The vehicle that contains the sensors that perceive the environment around the vehicle. https://se.mathworks.com/help/driving/ug/coordinate-systems.html (17.02.2025)

9. **/vehicle-vehicle_status** - Information on the status of the vehicle and its components.

These topics were selected to ensure the dataset effectively represented the vehicle's operational state, planned movements, and environmental perception.

Data Conversion (Task A.2.2)

The main outcome of Task A.2.2 was the conversion of ROS bag files into CSV format for easier use in machine learning. Each selected topic (as described in Task A.2.1) was extracted and saved as a separate CSV file, resulting in nine CSV files per test drive.

This was achieved using a Python program developed by Ali Ihsan Güllü (PhD student, University of Tartu), which processed the ROS bag files and structured the data for further processing. In total, 27 test drives (from Step A.1) were converted, ensuring a consistent and accessible format for model training.

Data Processing Pipelines (Task A.2.3 to Task A.2.5)

The key result of the set of Tasks A.2.3 to A.2.5 was a fully processed dataset, integrating vehicle control, localization, disengagement events, and environmental perception data into a structured format for model training. This was achieved through a series of preprocessing steps that merged, cleaned, and refined the data.

All preprocessing scripts used in these steps can be found in the repository: GitHub Repository. Below is an overview of each step and its outcome:

Combining the Data (Task A.2.3)

- **Objective:** Combine selected topics into a single CSV file per test drive, aligning data by timestamps.
- **Key Processing steps:**
    - Conversion of nested JSON data (detected objects, planned waypoints) into structured CSV format.
    - Forward and backward filling to handle missing values.
    - Removal of irrelevant metadata to streamline the dataset.
- **Output**: A structured dataset with data from all nine selected topics listed in Task A.2.1.
    - An excerpt of a sample output can be found in the following file: "2023-10-16-10-14-49_tiksoja_ride_02_sfa_combined_topics.csv" (GitHub Repository)

Labelling the Disengagement Events (Task A.2.4)

- **Objective:** Enhance the dataset with manually labeled disengagement events from ADL.
- **Key Processing steps:**
    - Addition of disengagement-related columns (event occurrence, reason, and type).
    - Standardization of labels for consistency in model training.
- **Output**: A dataset enriched with disengagement events and their key characteristics.
    - An excerpt of a sample output can be found in the following file: "2023-10-16-10-14-49_tiksoja_ride_02_sfa_expanded.csv" (GitHub Repository)

Extracting the Disengagement Events (Task.A.2.5)

- **Objective:** Create event-specific datasets by extracting five-second time windows preceding each disengagement event.
- **Key Processing steps:**
    - For each disengagement event, isolate a five-second window leading up to the switch from autonomous to manual driving.
    - Save each extracted timeframe as a separate CSV file for model training.
- **Rationale for the Five-Second Window**: Through testing, five seconds provided the optimal balance between capturing sufficient pre-disengagement context and managing RAM usage efficiently. It saved exactly enough memory while still ensuring reliable model performance, making it a practical choice over other tested durations (3-10 seconds).
- **Output**: 160 CSV files, each representing a disengagement event with its preceding driving data.
    - A sample output can be found in the following file: "2023-10-16-10-14-49_tiksoja_ride_02_sfa_disengagement_1.csv" (GitHub Repository)

Preparing the Data for Model Training (Task A.2.6)

The key result of Task A.2.6 was a fully preprocessed, augmented, and balanced dataset of 2,560 disengagement event dataframes, split into training and validation sets for model training. A sample of this fully processed data can be found in the file "sample_X_train.csv" (input data) and the corresponding label in "sample_y_train.csv" (GitHub Respository). This

sample represent the outcome after normalization, padding, and balancing, providing a clear example of the data structure used for model training.

As outlined in Task A.2.5, the full dataset initially included 160 disengagement events, which were used for both training and testing purposes. For model training, 6 events were reserved for testing, selected from classes 0 to 6 (excluding class 5). This left 154 disengagement events for the training set. These 6 test events will be revisited in Phase B when the test set is constructed.

After separating the training and test sets, both sets were normalized separately to ensure consistent feature scaling. The initial class distribution for the training set was highly imbalanced (see Table 4). To address this, data augmentation was applied to the training set, generating additional samples to balance the dataset. Each class in the training set was augmented to contain 320 samples. This number was chosen based on the typical requirement for a few hundred samples to train robust models, while also considering potential memory constraints due to the large size of the data. Finally, both the training and test sets were padded to ensure a fixed length for consistency.

Table 4. The count of disengagement classes for training after preprocessing.

Disengagement class	**Count (unbalanced)**	**Count (after balancing)**
0	64	320
1	29	320
2	13	320
3	23	320
4	12	320
6	11	320
7	1	320
8	1	320
SUM	**154**	**2560**

For each binary classification model:

- **320 samples** from the target class were labeled as 1 [True].
- **320 samples were randomly selected from the other classes**, ensuring an equal number of samples from each, and were labeled as 0 [False].

- The final dataset (640 samples per model) was split 80/20 while maintaining class balance:
    - **Training set (80%)**: 256 samples from class 1 [True], 256 from class 0 [False].
    - **Validation set (20%)**: 64 samples from class 1 [True], 64 from class 0 [False].

This preprocessing step ensured that the dataset was balanced across classes and followed a uniform distribution for the selection of samples, enabling the training of robust classification models.

Step A.3: Constructing the Classification Models

Step A.3 resulted in eight binary classification models based on a CNN architecture, as described in Sections 4.1 and 4.1.1. The models were built (Task A.3.1) using the specified layers, optimizer, loss function, and additional techniques such as EarlyStopping and class weighting, following the Methodology section.

For training (Task A.3.2):

- **100 epochs** were used, with EarlyStopping typically activating between 70-90 epochs, preventing unnecessary computation while ensuring sufficient training.
- **A batch size of 8** was chosen to enable frequent weight updates, as larger values led to increased validation loss.

This structured approach ensured optimal model performance while maintaining computational efficiency.

4.2 Phase B: Evaluation of the Classification Models

The primary results of Phase B were the confusion matrices and precision metrics, which measured the performance of the classification models on the test set. The following sections outline the key steps from Phase B as described in the Methodology, along with their corresponding results.

Step B.1: Constructing the Test Set

The main intermediate result of Step B.1 was the test set, which was identical for all models to ensure a consistent evaluation. It consisted of 30 samples, with five samples per class for classes 0 to 6.

Preprocessing for the test set (Task A.2.6) involved normalization and padding to ensure consistent feature scaling and fixed-length data, aligning with the preprocessing applied to the training set. However, the key focus of Step B.1 was data augmentation, which expanded the dataset from 6 to 30 samples.

To construct the test set, one original sample per class (extracted in Task A.2.6) was selected. Data augmentation was then applied to each of these original samples, generating five additional samples per class, resulting in a total of 30 test samples. Classes 5 ("Bad Engage") and 7 and 8 were excluded from the test set, as they lacked sufficient original samples for meaningful evaluation.

A sample of this fully processed data can be found in the file "sample_X_train.csv" (input data) and the corresponding label in "sample_y_train.csv" (GitHub Respository). While no separate test set sample is provided, its structure is identical to the training samples; thus, "sample_X_train.csv" can serve as a reference for understanding the format of test data, which follows the same preprocessing and structural conventions.

Step B.3: Evaluating Model Performance

The main results of Step B.3 were the confusion matrices and the corresponding performance metrics, which summarize how well the models classified the test samples. These results were derived from the models' predictions on the 30 test samples, ensuring a consistent evaluation framework across all models.

Each model's predictions were compared against the true labels to compute true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These outcomes are summarized in Table 5, which provides a detailed breakdown of classification results per sample.

For each sample, probability scores were recorded for all class labels. If multiple models assigned different labels to a sample, it was marked as "Uncertain." If no model assigned a probability above 0.5, the label was recorded as "Undetermined." Table 6 presents these probability scores and the final assigned labels.

Color Coding:

- **Green**: Correct predictions (True Positives & True Negatives)
- **Red**: Incorrect predictions (False Positives & False Negatives)

It turned out that no FP and FN cases occurred. Therefore, all cells in Table 5 are green.

Table 5. The prediction outcomes for all 30 test samples, showing TP, TN, FP, FN for each prediction along with the true label (TL) and the final assigned label (FL).

ID	TL	CL0	CL1	CL2	CL3	CL4	CL6	CL7	CL8	FL
1	0	**TP**	TN	TN	TN	TN	TN	TN	TN	0
2	0	**TP**	TN	TN	TN	TN	TN	TN	TN	0
3	0	**TP**	TN	TN	TN	TN	TN	TN	TN	0
4	0	**TP**	TN	TN	TN	TN	TN	TN	TN	0
5	0	**TP**	TN	TN	TN	TN	TN	TN	TN	0
6	1	TN	**TP**	TN	TN	TN	TN	TN	TN	1
7	1	TN	**TP**	TN	TN	TN	TN	TN	TN	1
8	1	TN	**TP**	TN	TN	TN	TN	TN	TN	1
9	1	TN	**TP**	TN	TN	TN	TN	TN	TN	1
10	1	TN	**TP**	TN	TN	TN	TN	TN	TN	1
11	2	TN	TN	**TP**	TN	TN	TN	TN	TN	2
12	2	TN	TN	**TP**	TN	TN	TN	TN	TN	2
13	2	TN	TN	**TP**	TN	TN	TN	TN	TN	2
14	2	TN	TN	**TP**	TN	TN	TN	TN	TN	2
15	2	TN	TN	**TP**	TN	TN	TN	TN	TN	2
16	3	TN	TN	TN	**TP**	TN	TN	TN	TN	3
17	3	TN	TN	TN	**TP**	TN	TN	TN	TN	3
18	3	TN	TN	TN	**TP**	TN	TN	TN	TN	3
19	3	TN	TN	TN	**TP**	TN	TN	TN	TN	3
20	3	TN	TN	TN	**TP**	TN	TN	TN	TN	3
21	4	TN	TN	TN	TN	**TP**	TN	TN	TN	4
22	4	TN	TN	TN	TN	**TP**	TN	TN	TN	4
23	4	TN	TN	TN	TN	**TP**	TN	TN	TN	4
24	4	TN	TN	TN	TN	**TP**	TN	TN	TN	4
25	4	TN	TN	TN	TN	**TP**	TN	TN	TN	4
26	6	TN	TN	TN	TN	TN	**TP**	TN	TN	6
27	6	TN	TN	TN	TN	TN	**TP**	TN	TN	6
28	6	TN	TN	TN	TN	TN	**TP**	TN	TN	6
29	6	TN	TN	TN	TN	TN	**TP**	TN	TN	6
30	6	TN	TN	TN	TN	TN	**TP**	TN	TN	6

Table 6. The results for all 30 test samples, showing the true label (TL), predicted label (FL), and probability score for each sample.

ID	TL	CL0	CL1	CL2	CL3	CL4	CL6	CL7	CL8	FL
1	0	**0.6879**	0.0035	0.0000	0.0000	0.0021	0.0007	0.0000	0.0000	0
2	0	**0.7158**	0.0043	0.0000	0.0000	0.0020	0.0006	0.0000	0.0000	0
3	0	**0.7414**	0.0053	0.0000	0.0000	0.0019	0.0006	0.0000	0.0000	0
4	0	**0.7454**	0.0055	0.0000	0.0000	0.0018	0.0006	0.0000	0.0000	0
5	0	**0.7872**	0.0079	0.0000	0.0000	0.0016	0.0006	0.0000	0.0000	0
6	1	0.0000	**0.8231**	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000	1
7	1	0.0000	**0.8686**	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000	1
8	1	0.0000	**0.8433**	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000	1
9	1	0.0000	**0.8596**	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000	1
10	1	0.0000	**0.8028**	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000	1
11	2	0.0000	0.0000	**0.9982**	0.0000	0.0000	0.0000	0.0000	0.0000	2
12	2	0.0000	0.0000	**0.9980**	0.0000	0.0000	0.0000	0.0000	0.0000	2
13	2	0.0000	0.0000	**0.9983**	0.0000	0.0000	0.0000	0.0000	0.0000	2
14	2	0.0000	0.0000	**0.9982**	0.0000	0.0000	0.0000	0.0000	0.0000	2
15	2	0.0000	0.0000	**0.9982**	0.0000	0.0000	0.0000	0.0000	0.0000	2
16	3	0.0000	0.0008	0.0000	**0.9982**	0.0000	0.0000	0.0000	0.0000	3
17	3	0.0000	0.0008	0.0000	**0.9982**	0.0000	0.0000	0.0000	0.0000	3
18	3	0.0000	0.0006	0.0000	**0.9987**	0.0000	0.0000	0.0000	0.0000	3
19	3	0.0000	0.0012	0.0000	**0.9974**	0.0000	0.0000	0.0000	0.0000	3
20	3	0.0000	0.0011	0.0000	**0.9975**	0.0000	0.0000	0.0000	0.0000	3
21	4	0.0000	0.0003	0.0000	0.0000	**0.9999**	0.0000	0.0000	0.0000	4
22	4	0.0000	0.0003	0.0000	0.0000	**0.9999**	0.0000	0.0000	0.0000	4
23	4	0.0000	0.0002	0.0000	0.0000	**0.9999**	0.0000	0.0000	0.0000	4
24	4	0.0000	0,0003	0.0000	0.0000	**0.9999**	0.0000	0.0000	0.0000	4
25	4	0.0000	0.0002	0.0000	0.0000	**0.9999**	0.0000	0.0000	0.0000	4
26	6	0.0000	0.0001	0.0000	0.2878	0.0000	**0.9613**	0.0000	0.0000	6
27	6	0.0000	0.0000	0.0000	0.3020	0.0000	**0.9745**	0.0000	0.0000	6
28	6	0.0000	0.0001	0.0000	0.2896	0.0000	**0.9639**	0.0000	0.0000	6
29	6	0.0000	0.0003	0.0000	0.2846	0.0000	**0.9431**	0.0000	0.0000	6
30	6	0.0000	0.0002	0.0000	0.2852	0.0000	**0.9542**	0.0000	0.0000	6

Based on the prediction outcomes, confusion matrices were constructed for each model to visualize classification performance. Standard evaluation metrics such as precision, recall, F1-score and accuracy were derived from these matrices to assess model effectiveness. Table 7 presents the confusion matrix for models CL0 to CL6, which share the same matrix. For models CL7 and CL8, the corresponding confusion matrix is shown in Table 8.

Table 7. The confusion matrix for models CL0 to CL6.

	Actual Positive	Actual Negative
Predicted Positive	5 (TP)	0 (FP)
Predicted Negative	0 (FN)	25 (TN)

Table 8. The confusion matrix for models CL7 and CL8.

	Actual Positive	Actual Negative
Predicted Positive	0 (TP)	0 (FP)
Predicted Negative	0 (FN)	30 (TN)

The confusion matrices indicate that all models performed flawlessly on the test set, with no false positives or false negatives. Each model correctly classified all 30 test samples, with no instances labeled as "uncertain" or "undetermined." Consequently, the performance metrics - accuracy, precision, recall, and F1-score - were all 100% for each model individually and collectively. These results demonstrate the models' exceptional ability to classify disengagement events with perfect accuracy on the test set.

4.3 Phase C: Explaining the Classification Models

Phase C section presents the SHAP value analysis, which provides insights into the models' decision-making processes. Notably, analyzing the confusion matrices did not yield additional valuable information, as all test samples were classified correctly, resulting in no false positives or negatives. This suggests that the dataset was well-separated for the classification task, with no systematic misclassification patterns. Consequently, prediction outcome analysis (Step C.1) did not allow for meaningful model comparisons.

To better understand why the models performed so well, Explainable Artificial Intelligence (XAI) methods were used (Step C.2). For each model, SHAP class **shap.DeepExplainer** [13] was used to generate SHAP values. This class was meant to approximate SHAP values for the deep learning models that were created, and it explicitly used the models' internal gradients when computing the values. As the dataset used had 484 features, not all could be efficiently analyzed. Therefore, to enhance the interpretability, SHAP values were processed as follows:

- **Feature Aggregation**: Top five features per model were extracted separately from both most positive and negative influences.
- **Feature Normalization:** Feature names with numerical suffixes were standardized (replacing numbers with "_i") to ensure consistency. If multiple features had the same base name with different suffixes, only the first occurrence was retained for analysis.
- **Value scaling:** SHAP values were scaled by a factor of one million to improve readability in visualizations, without altering their relative importance.
- **Sorting**: Feature names and model numbers were sorted alphabetically to ensure the visualization was structured.

The heatmap in Figure 6 provides a comparative overview of feature importance across different models, summarizing how various features influenced predictions. The key components of the visualization include:

- **Color schema**:
    - **Red Shades** indicate positive SHAP values, meaning that the feature increased the likelihood of a particular class being predicted.
    - **Blue Shades** indicate negative SHAP values, meaning that the feature decreased the likelihood of a particular class being predicted.
    - **White** represents that the features were not in the top 5 list of features that most positively or negatively impacted the predictions.
- **Columns**: Each column represents a feature (normalized).
- **Rows**: Each row represents a specific model.
- **Annotations**: The numeric values represent the scaled SHAP values.

[13] Shap.SeepExplainer documentation. https://shap.readthedocs.io/en/latest/generated/shap.DeepExplainer.html (14.02.2025)

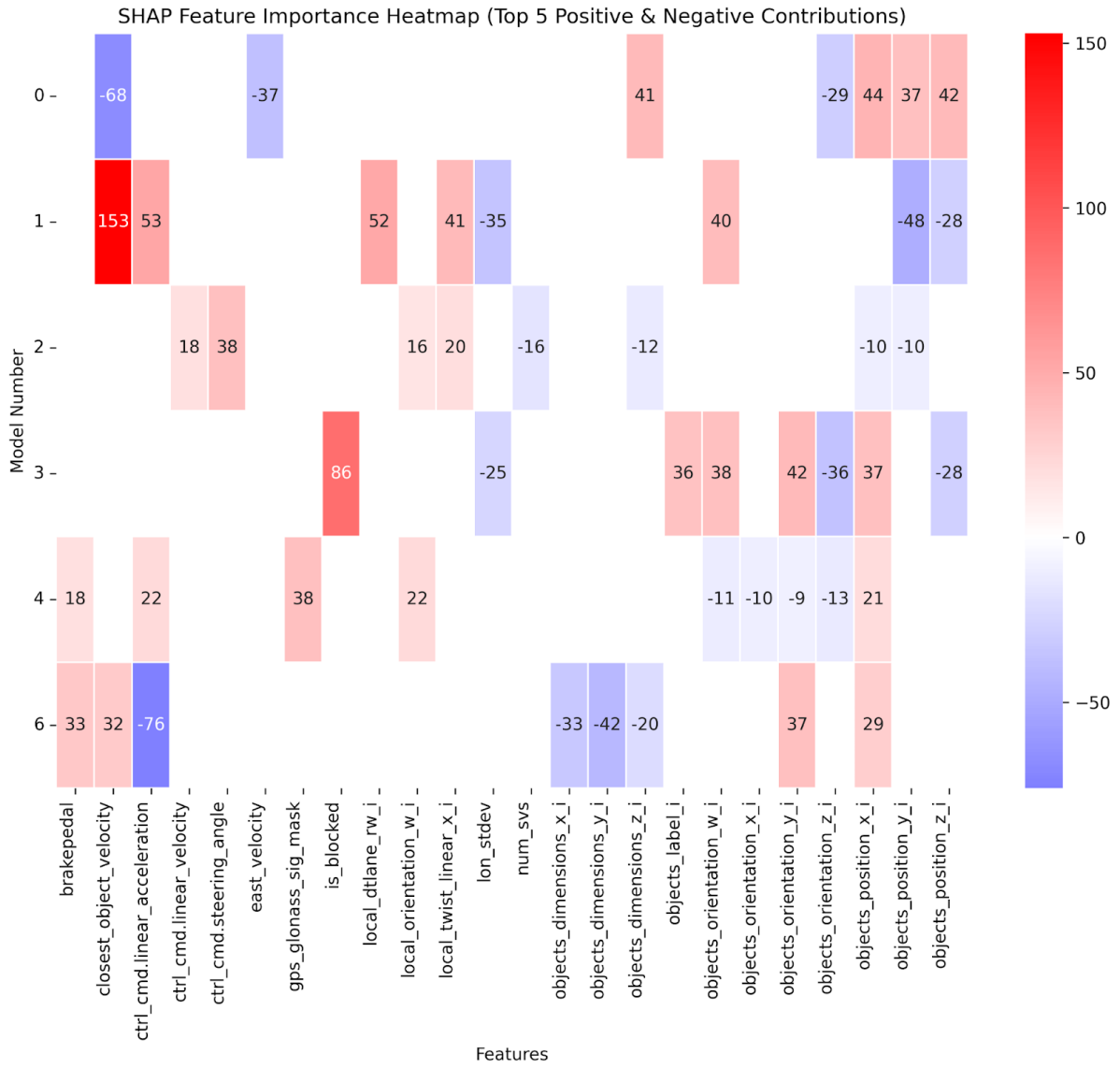

Figure 6. Visualization of the features that most positively and most negatively impacted the predictions per model.

The heatmap provided valuable insights into each model's decision-making process and allowed for direct comparisons between models:

- The larger the SHAP value and the darker the red shade, the more strongly a feature contributed positively to the model's prediction. In other words, these features increased the likelihood of the model assigning a given class.
- Conversely, the smaller the SHAP value and the darker the blue shade, the more strongly a feature contributed negatively to the prediction, meaning the feature actively discouraged the model from assigning that class.

Ideally, each model should focus on a distinct set of features. If the heatmap shows disjoint sets of red and blue shades for each model, it indicates that the models are utilizing different feature subsets for classification.

4.3.1 Per-Model SHAP Analysis

SHAP value analysis was performed separately for each model, with a focus on the distinct sets of positively (red) influencing features. This analysis helped interpret each model's behavior and assess whether the most influential features aligned with human intuition.

For instance, in the **Class 0** model (which classifies disengagements caused by obstacles), the most positively influential features were related to object detection, particularly position and dimension data. The four most influential features for Class 0 were (see red cells in the first row of Figure 6):

- **objects_position_x_i, objects_position_y_i, objects_position_z_i,**
- **objects_dimensions_z_i.**

These features highlight the model's reliance on spatial awareness, emphasizing the importance of nearby objects in classifying disengagements caused by obstacles blocking the vehicle's path. This focus on nearby objects aligns with human intuition, where detecting obstacles involves considering both their position and characteristics.

The heatmap can also be analyzed in terms of the shades of red (positive influence) and blue (negative influence). For instance, the **Class 1** model (which classifies disengagements caused by safety reasons) places significant emphasis on the closest object's velocity, indicating that nearby objects moving quickly can trigger safety-critical situations. In contrast, the **Class 0** model shows this feature as having a strong negative impact, suggesting that it did not focus on moving objects, which makes sense because obstacles tend to be static.

By analyzing the set of features for each model, valuable insights can be gained into which features played the most significant role in decision-making, how drastically they influenced the predictions, and whether these features align with human intuition.

A full analysis for each individual model is provided in **Appendix I**.

4.3.2 Comparative SHAP Analysis

Following the per-model analysis, the models were compared to assess how they performed as a set and how they differed from one another.

Ideally, each pair of models should focus on disjoint sets of both positive and negative features, indicating that they rely on different decision-making processes. For example, comparing **Class 0** (Obstacle disengagement) and **Class 1** (Safety disengagement) models, it was observed that their influential features were completely disjoint:

- None of the red-colored features (positive contributors) in Model 0 appeared as positive contributors in Model 1, and vice versa.
- Similarly, none of the blue-colored features (negative contributors) in Model 0 appeared as negative contributors in Model 1, and vice versa.
- Features that positively influenced one model often had the opposite effect in another, reinforcing that each model relied on distinct decision-making patterns.

However, feature separation was not always absolute across all models. In some cases, models had overlapping positive features but distinct negative ones, or vice versa. The degree of overlap varied:

- **Positive Feature Overlap**: The highest overlap was **50%**, observed between **Class 3** and **Class 6**, where two out of four key positive features were shared. In most other cases, the overlap was lower.
- **Negative Feature Overlap**: The highest overlap was **66%**, found between **Class 1** and **Class 3**, though for most other pairs, the overlap was minimal or nonexistent.

The full comparative analysis is provided in **Appendix II**.

These results suggest that while each model was highly specialized, they still shared some degree of commonality in feature selection. By leveraging mostly unique feature sets, the models were able to effectively differentiate between various disengagement scenarios.

Analyzing model decision-making using SHAP values provided valuable insights into why the models performed so well, why there were no misclassifications, what drove each model's predictions, and how the models differed from one another. While this provides a technical explanation of model behavior, some aspects may not be immediately intuitive. The Discussion

chapter will further explore how these results align with human intuition and real-world expectations.

5. Discussion

This section presents a summary of the model's performance, compares the approach to previous work, and discusses potential areas for improvement.

5.1 Overview of the Results

The methodology developed in this thesis represents a promising step toward fully automating the classification of disengagement events. The results demonstrate that each trained model (classes 0 to 8) performed exceptionally well, achieving 100% combined accuracy on the test set. Each model correctly classified test samples belonging to its respective disengagement class, with no false positives (FP) or false negatives (FN). Even for models 7 and 8, which had insufficient data for testing, 100% accuracy was maintained, as no incorrect classifications were recorded across all 30 test samples.

At first glance, the absence of any misclassifications might seem unexpected or even surprisingly ideal. However, an analysis using SHapley Additive exPlanations (SHAP) helps to clarify why the models performed so well. The SHAP values revealed that each model relied on a distinct set of highly correlated features, making misclassification unlikely. The fact that such a clear distinction emerged is a valuable insight that enhances our understanding of automated disengagement classification and could inform future model development.

The SHAP analysis also provided deeper insights into the logical reasoning behind each model's decision-making process. The key findings regarding feature importance for each class include:

- **Model 0 ("OBS")**: Focused on different aspects of detected objects, suggesting obstacle identification.
- **Model 1 ("Safety")**: Emphasized object speed, vehicle acceleration, and planned waypoints, indicating possible safety concerns such as unstable driving, path deviations, or nearby moving vehicles.
- **Model 2 ("Turnback")**: Focused on the steering movements of the vehicle, indicating turnback maneuvers.
- **Model 3 ("Pedestrian crossing")**: Prioritized detected objects labeled as "Pedestrian" and signals indicating road blockage, demonstrating pedestrian identification.
- **Model 4 ("Localization"):** Relied on GPS signals and vehicle movement data, indicating potential localization issues or deviations from the planned path.

- **Model 6 ("Give way")**: Considered detected objects, their features, and vehicle braking or speed adjustments, suggesting interactions at intersections with nearby vehicles.

For a comprehensive breakdown of the most important features in each class, along with their correlation to the disengagement categories, refer to **Appendix I**. There, the full per-model SHAP analysis provides further context on how these features align with human intuition and influence model predictions.

For models 7 and 8, SHAP analysis was not performed because there were no test samples from these classes, and consequently, the models did not classify any samples into their respective categories. Since SHAP explains a model's predictions by analyzing the contributions of different features, its output is only meaningful when there are positively classified instances. Without any positive classifications, SHAP would only highlight negative contributions - why samples were not classified into these classes - which would not provide meaningful insights into the actual feature importance for these disengagement types.

These findings from the SHAP analysis confirm that the selected features were both logically and technically meaningful. Logically, the models emphasized features that align with human intuition - each model prioritized data most relevant to its respective disengagement reason. Technically, the differences in feature importance across models indicate that each model's decision-making process was distinct, reinforcing that they were not simply learning generic patterns but rather capturing class-specific behaviors effectively.

Another key observation was how model confidence varied across classes. Models trained on classes with a higher number of original samples exhibited lower confidence, suggesting that these situations occur under diverse conditions, making it harder to identify clear patterns. Conversely, models trained on classes with fewer original samples mostly demonstrated higher confidence, implying that those situations tend to occur under more specific and consistent conditions.

5.2 Comparison with Previous Work

The result of this thesis can be compared to Jõelo's Bachelor's thesis "Automating Classification of Disengagements using Foxglove" conducted in 2024 [7]. Her work proposed a semi-automated methodology for classifying disengagements, distinguishing between planned and unplanned disengagements, and further sub-classifying planned disengagements.

Her approach, titled as the Disengagement Classifier Method (DCM), achieved:

- 75% precision and accuracy, and 100% recall in classifying planned disengagements into sub-classes.

In contrast, this thesis developed a fully automated classifier capable of sub-classifying both planned and unplanned disengagements. The methodology used a balanced test set, meaning the test set focused on having an equal number of samples from each class, rather than testing on ride-based disengagement events. The models of this thesis reached 100% accuracy, precision, recall and F1-score, suggesting that a neural network approach is more effective than a rule-based approach for the task at hand.

The following key differences can be brought out between the two approaches:

- **Data Format:** The DCM operates directly on .bag files within Foxglove, whereas this thesis uses Python-based Jupyter Notebook scripts and processes data in .csv format.
- **Preprocessing Requirements:** The DCM requires no additional preprocessing, while the methodology in this thesis involves data conversion and multiple preprocessing steps.
- **Sub-Classification Differences for Planned Disengagement Events:**
    - **DCM:** “Pedestrian Crossing”, “Temporary Roadwork”, “Bus Stop”, “Turnback”, “Give Way”.
    - **This Thesis:** “Pedestrian Crossing”, “Turnback”, “Give Way”, “STOP”, “SPEED”.
- **Classification Scope:** The DCM classifies only planned disengagements, grouping all unplanned cases into a single category. This thesis classifies both planned and unplanned disengagements into distinct classes.
- **Flexibility:** The DCM follows a rule-based (hardcoded) approach, while the neural network models proposed in this thesis are easily adaptable - new features can be incorporated, and retraining allows for integration with different software.

Overall, the fully automated approach proposed in this thesis offers better performance and adaptability. However, a key limitation is that it requires more preprocessing, which can be time-consuming, and could be prevented by working directly with the ROS .bags like the DCM.

5.3 Areas for Improvement

The primary limitation of this thesis was the availability of raw data. The dataset was selected to match the one used in Jõelo's thesis [7], as it had already been carefully analyzed by the ADL. Even though, the augmenting of the original samples provided a great approach for increasing the dataset size, more original samples would still be needed to enhance models' reliability. Especially problematic were classes 7 and 8, with only a single original sample for both classes, making proper training and testing for those categories difficult. Therefore, for the proposed methodology to be fully taken into use, training on more samples and additional testing is suggested.

Additionally, improving the preprocessing pipeline would make the approach more effective. The steps proposed in the methodology were separated into multiple scrips to make the understanding of the steps more intuitive, but this process could be improved by using a single script file, rather than relying on multiple ones. Additionally, the scripts used in this thesis use hard-coded topic names and file paths, which could be removed, making it more adaptable for different users, and the process of expanding the datasets with additional topics more flexible.

Currently, disengagements labeled as "Bad Engage" (Class 5) were replaced with the last seen label. However, removing this class entirely may not be ideal, as it holds relevance for post-drive analysis. A potential solution would be to introduce a script that detects new disengagements within a short time window (e.g., 10 seconds), preserving events labeled as "Bad Engage" while ensuring their proper classification.

6. Conclusion

This thesis aimed to develop a structured data preprocessing pipeline, implement a fully-automated classification system for disengagement events in autonomous driving, evaluate model performance, and provide insights into model decision-making process.

A fully automated classification system for disengagement events across eight classes in autonomous driving was successfully developed, achieveing 100% accuracy, precision, recall and F1-score across all classes. The SHAP analysis provided valuable interpretability, revealing how different features influenced classification decisions both at the individual model level and across the entire set of models. The neural network-based approach outperformed previously designed rule-based methods, demonstrating the effectiveness and flexibility of neural networks in this domain.

This research contributes to the advancement of post-drive analysis in autonomous driving and enhances safety by enabling accurate automated classification of disengagement events by reducing human workload and minimizing potential human errors.

Despite the models' success, limitations remain. The lack of sufficient original data, particularly for classes 7 and 8, restricted proper evaluation and testing. Additionally, the extensive data preprocessing required could introduce computational overhead.

Future research could focus on collecting more original disengagement samples to improve model reliability and quality, and ensure sufficient data for comprehensive testing and evaluation. The efficiency of the preprocessing pipeline could also be improved by consolidating multiple preparation steps into a single script or integrating the pipeline directly with ROS .bag files to streamline preprocessing as a whole.

Overall, this thesis represents a significant step toward fully automated disengagement classification and establishes a strong foundation for future advancements in post-drive analysis for autonomous driving.

Beyond post-drive analysis, this solution could serve as a valuable software evaluation tool by monitoring trends in disengagement events and their types over time. By automatically identifying how certain disengagement events fluctuate across different driving scenarios and software updates, developers can assess the effectiveness of the ADS software more efficiently.

This could enhance ADS development cycles and improve quality assurance by providing data-driven feedback on system performance.

List of References

[1] Chu M., Zong K., Shu Z., Gong J., Lu Z., Guo K., Dai X., Zhou G. Work with AI and Work for AI: Autonomous Vehicle Safety Drivers' Lived Experiences. 2023. https://doi.org/10.1145/3544548.3581564 (30.04.2025).

[2] Khan S. M., Salek M. S., Harris V., Comert G., Morris E. A. Chowdhury M., Autonomous Vehicles for All? *Journal on Autonomous Transportation Systems*, 2024, 1, 1. https://doi.org/10.1145/3611017 (30.04.2025).

[3] Karla N., Paddock S. M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 2016, 94. https://doi.org/10.1016/j.tra.2016.09.010 (30.04.2025).

[4] McGehee D. V., Brewer M., Schwarz C., Smith B. W. Review of Automated Vehicle Technology: Policy and Implementation Implications. 2016. https://rosap.ntl.bts.gov/view/dot/30702 (30.04.2025).

[5] Shokan A., Mareček J. Could Disengagement Reports Indicate Evolution of Autonomous Vehicles? *Vehicles*, 2025, 7, 32. https://doi.org/10.3390/vehicles7020032 (30.04.2025).

[6] Autonomous Driving Lab. https://adl.cs.ut.ee/ (30.04.2025).

[7] Jõelo P. Automating Classification of Disengagements Using FoxGlove. University of Tartu. Institute of Computer Science. Bachelor's thesis. 2024. https://thesis.cs.ut.ee/494db6fd-10d3-42ab-bd07-cee467d98893 (30.04.2025).

[8] SAE International. SAE Levels of Driving Automation Refined for Clarity and International Audience. 2021. https://www.sae.org/blog/sae-j3016-update (30.04.2025).

[9] ROS.org. ROS Introduction. 2018. https://wiki.ros.org/ROS/Introduction (30.04.2025).

[10] MathWorks. What Is an Autoencoder? https://se.mathworks.com/discovery/autoencoder.html (30.04.2025).

[11] GeeksforGeeks. What is a 1D Convolutional Layer in Deep Learning? 2024. https://www.geeksforgeeks.org/what-is-a-1d-convolutional-layer-in-deep-learning/ (30.04.2025).

[12] Gabor Petnehazi. Quantile Convolutional Neural Networks for Value at Risk Forecasting. 2020. https://doi.org/10.48550/arXiv.1908.07978 (30.04.2025).

[13] Pandas introduction. https://pandas.pydata.org/ (30.04.2025).

[14] Ma T., Zhou C. Selection of regularization model for linear regression under high-dimensional data. 2023. https://doi.org/10.1145/3613330.3613342 (30.04.2025).

[15] GeeksforGeeks. What is Batch Normalization In Deep Learning? 2025. https://www.geeksforgeeks.org/what-is-batch-normalization-in-deep-learning/ (30.04.2025).

[16] GeeksforGeeks. CNN | Introduction to Pooling Layer. 2025. https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (30.04.2025).

[17] Harjai A., Charan A., Singhal S. Sentiment Analysis of Medications Review Using Deep Learning Algorithms. 2024. https://doi.org/10.1145/3647444.3647828 (30.04.2025).

[18] Kingma D. P., Ba J. L. Adam: A Method for Stochastic Optimization. 2015. https://doi.org/10.48550/arXiv.1412.6980 (30.04.2025).

[19] Kulkarni C. A., Mohith R., Sharath S.S. Custom Binary Cross Entropy Based Anomaly Detection in Bank Transactions using Deep Convolutional Neural Network. 2021. https://doi.org/10.1145/3487664.3487708 (30.04.2025).

[20] Jhumka K., Auzine M. M., Khan M. H.-M., Casseem M. S., Fedally S. A., Mungloo-Dilmohamud Z. Explainable Chronic Kidney Disease (CDK) Prediction using Deep Learning and Shapley Additive Explanations (SHAP). 2024. https://doi.org/10.1145/3633598.3633604 (30.04.2025).

[21] Cunha B. M., Barbosa S. D. J. Evaluating the Effectiveness of Visual Representations of SHAP Values Toward Explainable Artificial Intelligence. 2024. https://doi.org/10.1145/3702038.3702093 (30.04.2025).

Appendices

Appendix I. Full Per-Model SHAP Value Analysis

This section presents a detailed analysis of the SHAP values for each model, highlighting the features that most positively impacted model predictions - features that the models identified as having the most significant influence on the classification of disengagement events. By focusing on these impactful features provides insight into the factors that most effectively shaped the model's decision-making process.

Class 0 - "OBS" (Obstacle) - Disengagement caused by obstacles.

For disengagements caused by obstacles obstructing the vehicle's path, the most influential features were related to object detection, including position and dimensions. The features that had the greatest impact on the model's performance were:

- **objects_position_x_i, objects_position_y_i, objects_position_z_i,**
- **objects_dimensions_z_i.**

These features highlight the model's ability to recognize the importance of nearby objects and their data, enabling it to accurately classify disengagements caused by obstacles obstructing the vehicle's path.

Class 1 - "Safety" (Safety Concerns) - Disengagement caused by safety considerations.

In scenarios where disengagements were triggered by safety concerns, such as objects getting dangerously close or abrupt changes in speed, the most significant features included the detected objects' velocity and orientation, along with the vehicle's linear acceleration and the planned waypoints it was following. The features that had the greatest impact on the model's performance were:

- **closest_object_velocity,**
- **ctrl_cmd.linear_acceleration,**
- **local_dtlane_rw_i,**
- **local_orientation_w_i.**

These features suggest that the model effectively captured situations where disengagements were related to potential safety risks.

Class 2 - “Turnback” (Turnback Maneuvers) - Disengagement caused by turnbacks.

For disengagements resulting from turnbacks, the model’s primary features revolved around the steering angle, the vehicle’s rotational movement (indicated by the twist of the planned path), and linear acceleration. The features with the greatest impact were:

- **ctrl_cmd.steering_angle,**
- **ctrl_cmd.linear_acceleration,**
- **local_orientation_w_i,**
- **local_twist_linear_x_i.**

This demonstrates that the model recognized turnbacks as involving significant changes in steering and rotational movement.

Class 3 - “Pedestrian Crossing” - Disengagement caused by pedestrians crossing the road.

In cases of disengagements due to pedestrians crossing, the most influential features included whether the vehicle was blocked (“is_blocked”) and details about detected objects such as their orientation, labels, dimensions, and positions. The features with the greatest impact were:

- **is_blocked,**
- **objects_label_i,**
- **objects_orientation_w_i, objects_orientation_y_i,**
- **objects_position_x_i.**

These findings show that the model effectively identified the importance of detecting and labeling pedestrians as critical for disengagements caused by pedestrian crossings.

Class 4 - “Localization” (Localization Issues) - Disengagements caused by problems with localization.

For disengagements caused by localization issues (such as insufficient satellite data or the vehicle straying from its planned path), the most important features were related to GPS data, vehicle acceleration, and brake pedal input. The prominent features were:

- **gps_glonass_sig_mask,**
- **ctrl_cmd.linear_acceleration,**
- **brakepedal,**
- **local_orientation_w_i.**

These features reveal that the model recognized localization issues as being tied to GPS signals, vehicle movement (e.g. orientation), and pedal inputs.

Class 6 - "Give Way" (Yielding at Intersections) - Disengagement caused by "give way" situations at intersections.

For disengagements related to "give way" situations, where the vehicle must yield to other traffic at intersections, the model highlighted acceleration and the velocity, orientations, and positions of detected objects as the most influential features. The prominent features were:

- **brakepedal,**
- **closest_object_velocity,**
- **objects_orientation_y_i,**
- **objects_position_x_i.**

These insights indicate that the model understood give-way scenarios typically involve changes in vehicle speed and detection of approaching vehicles or objects.

In conclusion, the SHAP value analysis provided important insights into the features each model prioritized, helping to explain their performance. By identifying the most positively impacting sets of features for each class, the analysis shows that the models focused on different but relevant aspects of the data. This alignment with human intuition supports the models' accuracy and helps explain why they performed well.

Appendix II. Comparative SHAP Value Analysis

This section details the comparative SHAP value analysis performed on the 6 models to examine how different disengagement classes rely on distinct or overlapping features. The analysis quantifies the extent to which positively and negatively contributing features are shared between pairs of classes.

The following tables summarize the feature overlap for both positive and negative contributors. A low overlap suggests that the models rely on distinct decision-making patterns, while a higher overlap indicates that some disengagement classes may share underlying patterns in the data. Refer to Figure 6 in Section 4.3 for the SHAP value heatmap, which visually represents feature contributions across the models.

Table 9 presents the overlap of positively contributing features between the six disengagement classes. The number of positive features identified for each class is indicated in square brackets next to the class label. The values in the table represent the percentage of shared features between each pair of classes, calculated relative to the smaller feature set of the two.

Table 9. Overlap of positively contributing features.

+	**Class 0** [4]	**Class 1** [5]	**Class 2** [4]	**Class 3** [5]	**Class 4** [5]	**Class 6** [4]
Class 0 [4]		0	0	0.25	0.25	0.25
Class 1 [5]	0		0.25	0.2	0.2	0.25
Class 2 [4]	0	0.25		0	0.25	0
Class 3 [5]	0.25	0.2	0		0.2	0.5
Class 4 [5]	0.25	0.2	0.25	0.2		0.5
Class 6 [4]	0.25	0.25	0	0.5	0.5	

Table 10 presents the overlap of negatively contributing features between the six disengagement classes. Similar to the previous table, the number of negative features identified for each class is listed in square brackets. The values represent the percentage of shared negative features between pairs, calculated relative to the smaller feature set.

Table 10. Overlap of negatively contributing features.

-	**Class 0** [3]	**Class 1** [3]	**Class 2** [4]	**Class 3** [3]	**Class 4** [4]	**Class 6** [4]
Class 0 [3]		0	0	~0.33	~0.33	0
Class 1 [3]	0		~0.33	~0.66	0	0
Class 2 [4]	0	~0.33		0	0	0.25
Class 3 [3]	~0.33	~0.66	0		~0.33	0
Class 4 [4]	~0.33	0	0	~0.33		0
Class 6 [4]	0	0	0.25	0	0	

The results show that while some model pairs have completely disjoint feature sets, others share a limited number of positive or negative features. However, the overlap remains relatively small or nonexistent in most cases. The main observations include:

- **Minimal Overlap**: Most class pairs exhibit low feature sharing, suggesting that the models differentiate disengagement classes using distinct patterns.
- **Instances of Higher Overlap**: A few class pairs (e.g., Class 3 and Class 6 for positive features and Class 1 & Class 3 for negative features) show higher overlap, indicating potential similarities in how these disengagement events are characterized.

The limited feature overlap reinforces that the models effectively distinguish disengagement types while allowing for some shared decision-making elements where disengagement reasons might have similar patterns in the data.

This comparative analysis provides deeper insights into how the models differentiate disengagement events and highlights the degree of feature uniqueness across classes.

License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Elisabet Hein,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

   Automating the Classification of Disengagements Using Convolutional Neural Networks,

   supervised by Dietmar Alfred Paul Kurt Pfahl;

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Elisabet Hein

14/05/2025