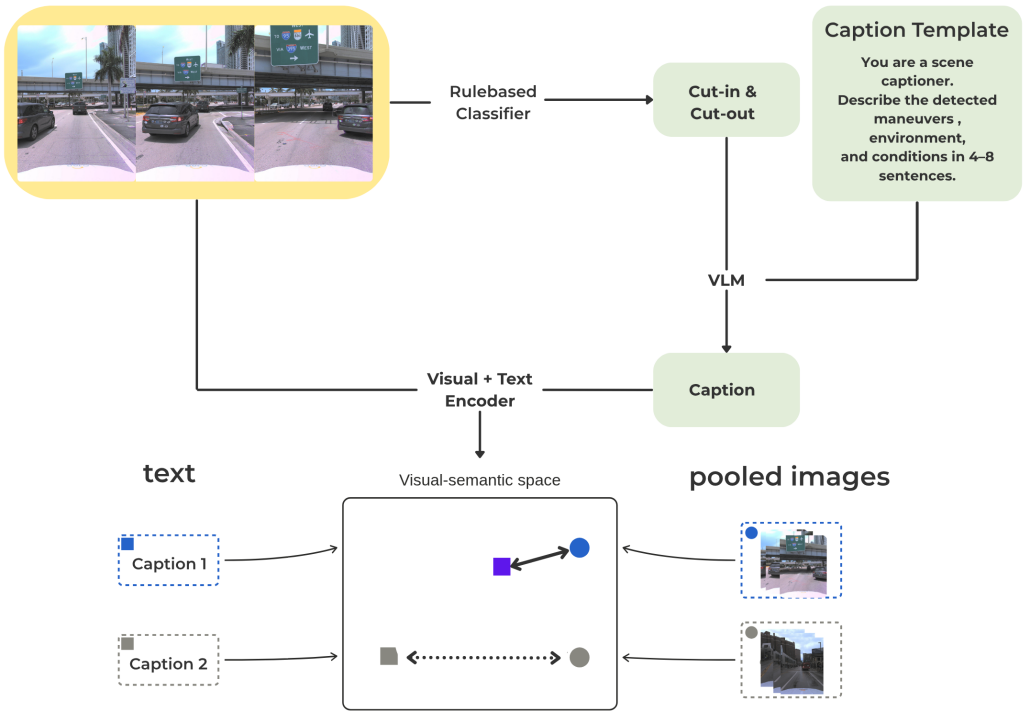
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Captioning Engine for AD/ADAS Data using Multi-Modal Large Language Models

Master's thesis in Data Science and AI

Noah Johnsson and Marcus Müntzing

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Captioning Engine for AD/ADAS Data using Multi-Modal Large Language Models

Noah Johnsson and Marcus Müntzing

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Captioning Engine for AD/ADAS Data using Multi-Modal Large Language Models
Noah Johnsson and Marcus Müntzing

Supervisor: Ali Nouri, Computer Science and Engineering
Advisor: Felix Rosberg, Zenseact
Examiner: Dag Wedelin, Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Captioning Engine for AD/ADAS Data using Multi-Modal Large Language Models
Noah Johnsson and Marcus Müntzing
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS) are dependent on perception models that can understand and interpret their environment and surroundings. These models require a large amount of annotated data to achieve this goal. However, it is often difficult to curate large datasets as it is time-consuming, most footage is often redundant, and it is hard to find the rare edge cases that matter the most. Vision-language models (VLMs) offer a scalable alternative to automate the process of generating captions for driving scenarios. This thesis investigates how such captions can be used through representation learning for downstream tasks.

First, a captioning engine combines rule-based detection of driving maneuvers (lane changes, cut-ins and cut-outs) with a VLM to generate detailed captions describing the environment and interactions in the scene. The rule-based component compensates for the VLM's limited ability to reason about temporal progression across frames, while the VLM contributes rich descriptions of the surrounding scene.

Secondly, the resulting video and caption pairs are used to fine-tune two contrastive vision-language embedding models, CLIP (ViT-L/14) and the Perception Encoder (L14-336), with the goal to align the captions and videos in a shared representation space. To our knowledge, this is the first use of the Perception Encoder in the AD/ADAS domain.

Our results show that a rule-based classifier on dash-cam sequences could be effectively used as context enrichment of the caption generating VLM. Fine-tuning on these detailed and rich captions increases text-to-video Recall@1 from 25% to 83%. CLIP outperforms the Perception Encoder as the choice of backbone across all metrics and all maneuver types after fine-tuning.

Further testing is needed to evaluate the generalization of this method beyond the maneuvers and dataset considered here. Future work includes extending the rule-based system to additional maneuvers, such as lead-vehicle braking and vulnerable-road-user interactions, and evaluating whether the learned representations transfer to other datasets.

Keywords: AD, ADAS, VLM, contrastive vision-language embedding models, Perception Encoder, captioning engine, PE, CLIP

Acknowledgements

We would like to express our gratitude towards our supervisor Felix Rosberg at Zenseact, for his valuable feedback and guidance throughout this project. We would also like to thank Rickard Persson and Basile Vermassen at Zenseact for their support. Finally, we thank our advisor Ali Nouri at Chalmers and our examinator Dag Wedelin for their advice during this project.

Noah Johnsson and Marcus Müntzing, Gothenburg, 2026-06-02

Contents

List of Figures

List of Tables

1

Introduction

Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS) are fundamentally dependent on high-quality perception models that can interpret inputs from various modalities such as 3D point clouds and HD map annotations [1]. Traditional computer vision tasks such as object detection and motion prediction have been used to understand these visual driving scenarios [2].

Modern AD datasets cover a wide range of traffic scenarios and edge cases, but extracting temporal and semantic understanding from this data remains challenging. In particular, Vision-Language Models (VLMs) still struggle to capture the temporal structure of driving scenes, limiting their usefulness for downstream tasks such as scenario retrieval and curation [2], [3].

Zenseact, an industrial partner in this thesis, develops perception and decision-making systems for automotive applications [4]. In such an industrial setting, efficient methods for organizing, searching, and curating collected data are essential. Natural-language captions are a good tool for this purpose, since they describe scenes at a higher semantic level than traditional annotations such as bounding boxes or trajectories, and allow scenarios to be retrieved through text queries. However, manually writing captions for large-scale driving datasets is costly and difficult to scale [2]. This has contributed to the work of using language models to automate the process of generating grounded and rich captions of driving scenarios [2].

Recent advances in VLMs have shown that using natural language on driving scenes enables a deeper semantic understanding of the environment than traditional vision-only models [5]. Models such as CLIP offer a scalable way to search and curate visual data through natural-language queries, since they embed images and text into a shared representation space [6]. A more recent model, the Perception Encoder, extends this paradigm to video, producing embeddings tuned for both image and video understanding and showing strong results on zero-shot classification tasks [7]. Within AD it can be utilized for curation of data and retrieval of corner cases [5]. Despite this potential, current models describe static scene attributes well but struggle to capture the dynamic actions and interactions that characterize driving scenarios [2].

This thesis aims to address these shortcomings by developing high-quality captions tailored to AD/ADAS data and by fine-tuning a contrastive vision-language embedding model so that they better represent driving scenarios. The resulting system

would support scalable data curation, improve edge case retrieval, and enable more efficient development of downstream perception models.

1.1 Problem Formulation

AD and ADAS development relies on rich understanding of driving scenarios across large data collections. General-purpose foundation models and captioning approaches provide a strong starting point, but specializing them for driving-scene retrieval introduces several challenges that motivate this work.

First, there is a lack of captions that are of high-quality and specific for the AD/ADAS domain. Several studies have explored the use of language-guided visual understanding of driving scenarios, either by enriching existing datasets with additional textual information, such as nuScenes-QA and DriveLM, or creating new datasets independently, such as BDD-X and DRAMA [1]. Despite these advances, existing dataset remain limited in scale and quality to address the challenges of autonomous driving [1]. Perception tasks that use language models for retrieving and understanding driving scenarios are thus still in an early stage [5]. As a result, current captions lack richness and are not grounded in temporal dynamics that is needed for downstream tasks.

Second, although video captioning models show strong performance on general datasets, their performance is not as good on driving scenarios [2]. Existing models are not as precise as needed to describe detailed actions in the scene, such as identifying the lane of a car or describing the agents relative to the ego vehicle [2]. For example, the models struggle to generate captions like "the ego vehicle is changing lane from the left lane to the right lane while the vehicle ahead is slowing down". This motivates the development of a detailed captioning engine that produces rich, contextual captions on which a vision-language encoder can be fine-tuned.

Another motivation for this work is that rich and grounded captions for driving data are often annotated by humans, which is time consuming and costly [2]. This motivates the use of VLMs to automate the process of generating annotations for datasets.

The work is also motivated by recent advances in contrastive learning. Existing work mainly uses CLIP to fine-tune their models [5]. As recent work on the Perception Encoder has shown, CLIP uses final-layer representations and therefore tends to lose important spatial information [7]. This is a limitation for driving scenarios, as understanding lane geometry, depth, and relative motion is needed for generate precise captions. The Perception Encoder addresses this by using intermediate representations to extract spatial information [7]. However, to the best of our knowledge, the Perception Encoder has not been used on autonomous driving data yet, which motivates the exploration of it.

Together, these limitations reveal a research gap. Previous work has shown the value of using rules to generate captions, for example, by training SwinBERT on LiDAR-derived templates [2]. However, such methods rely on the templates directly for

generating captions, and no existing approach combines a VLM and a contrastive encoder. This motivates the work of developing a captioning engine in which first a rule-based system identifies traffic maneuvers from annotated map and traffic data, and resulting descriptions are then giving to the VLM to generate rich and temporally grounded captions. These captions are then used to fine-tune a contrastive vision-language embedding model on the resulting video-caption pairs, addressing the limitations above.

1.2 Aim and Objectives

The overall aim of this thesis is to develop a scalable captioning and representation-learning framework for AD and ADAS data.

More specifically, the objectives are:

1. **Develop a captioning engine for driving scenarios**
   The first objective is to build a captioning engine for driving scenarios that produces rich and descriptive captions for videos. The engine uses rule-based methods to identify specific driving maneuvers, such as cut-ins, cut-outs, and lane changes, together with a VLM. The captions should not only describe the ego vehicle but also how surrounding traffic agents interact in the scene. This objective also includes evaluating the captions to assess the correctness, specifically whether the identified maneuvers and scene descriptions accurately reflect the video.

2. **Fine-tuning a vision-language embedding model using contrastive learning**
   The second objective is to use the captions from the captioning engine to fine-tune a contrastive vision-language embedding model such as CLIP and the Perception Encoder. By using the video-captions pairs as supervision, the goal is to learn representations of the videos that improve retrieval metric for driving scenarios.

1.3 Limitations

The scope of this thesis is limited in several ways. First, the Argoverse 2 dataset is exclusively used, which consists of driving data in six U.S. cities. The generalizability of the results to other cities and geographic regions, or other sensor setups is therefore not evaluated. Second, the captioning engine is restricted to capture three different driving events: lane changes, cut-ins, and cut-outs. Other maneuvers such as overtaking and braking are consequently outside this scope. Third, the finetuning of the contrastive encoder is limited to a small subset of 700 training sequences. Although augmentation and sliding window sub-sequences are implemented to increase the number of training samples, this may still limit the representational capacity of the resulting model. Fourth, this thesis compares two contrastive vision-language embedding models, CLIP and the Perception Encoder, and does not include compar-

isons against other types of architectures. Finally, computational constraints limit the scope of the fine-tuning experiments, as full encoder training and hyperparameter sweep were not feasible with given resources.

2

Theory

This section provides a background on ADAS and AD. It also introduces vision-language models and captioning engines, which form the technical foundation of this work.

2.1 Overview of ADAS and AD

ADAS and AD refers to a set of vehicle technologies that are designed to assist and support the driver and improve overall safety. The first ADAS systems included parking assistance and cruise controls, and as technology advanced, the focus shifted toward increased safety for both the driver and pedestrians. ADAS can have many positive impacts. For example, they can improve driver behavior through reduced driving speed and variability, enhanced alertness, and mitigation of exposure to risky conditions [8]. An overview of different use-cases is shown in Figure 1.

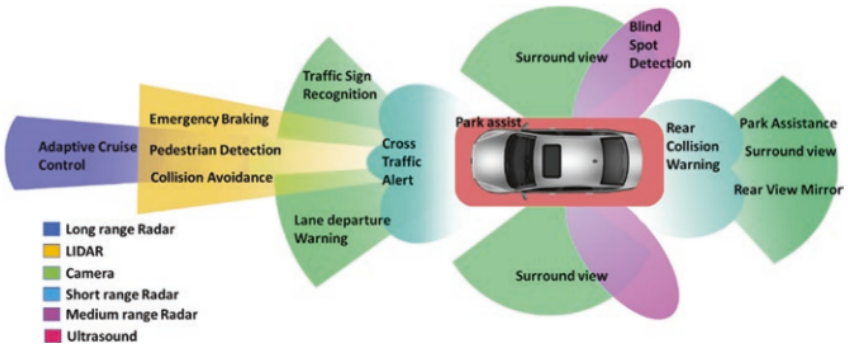


Figure 2.1: ADAS use cases [8].

There are different types of technologies used in ADAS, shown in table 2.1. The main sensors include camera vision, radio detection and ranging (RADAR), light detection and ranging (LiDAR), and ultrasound. Cameras act as the eyes of ADAS and are mainly used for applications such as traffic sign identification, parking assistance and lane departure warnings. A limitation with cameras is that they are strongly affected by darkness, low light, and poor weather conditions. RADAR, on the other hand, are good in any weather and are used for both short- and long-range applications. LiDAR measures distances using laser reflections and can provide

geometric information of objects around the vehicle. Lastly, ultrasonic sensors are used mostly in short-range applications and are based on the principle of sound to determine distances to nearby objects and warn the driver about these [8].

Table 2.1: Various sensors used in ADAS and their typical applications [8].

Sensor	**Maximum range (m)**	**Typical applications**
Short-range radar	20	Blind spot detection
Long-range radar	150	Adaptive cruise control (ACC)
Camera vision	80	Pedestrian detection, sign recognition, parking assistance, lane departure warning
Infrared	120	Night vision
Ultrasound	3	Park assist
LiDAR (scanning)	120	Emergency braking, collision avoidance systems

2.2 VLMs for ADAS and AD

Vision-Language Models (VLMs) bridge the gap between computer vision and natural language processing by learning a shared representation between visual data and natural language [5]. This joint representation enables models to retrieve both images and videos, answer questions based on visual input, and generate descriptive captions in natural languages [5]

The use of language models in the autonomous driving sector has increased over the last years. These models have gained in interest as they are able to analyze non-textual data through textual representations, leading to large improvements in zero-shot image classification and object detection [1]. In this context, VLMs could serve as the bridge to support human-machine interactions, potentially transforming how humans engage with vehicles by interpreting traffic scenes and improving driver decision-making [1].

Since these models are trained on a lot of diverse data, including different type of traffic scenarios and maps, their overall goal is to enhance vehicle planning and navigation to increase safety and efficiency [1]. Furthermore, interactions based on language that VLMs offer can help mitigate the black-box nature of the ADAS/AD-models by providing explanatory feedback that accompanies their decisions [9]. Large VLMs can also leverage their outstanding performance in zero-shot generalization to improve recognition abilities in corner cases and for previously unknown objects [5].

The importance of these models increases with higher levels of automation. The Society of Automotive Engineers classifies AD development on a scale from Level 0 (No Automation) to Level 5 (Full Automation) [1]. Currently, most commercially deployed systems are on level 2 and level 3, meaning that the system is partially

automated but still requires human supervision from the driver. As systems are moving to a higher level of automation perception becomes more important [5].

Perception modules form the foundation of any autonomous driving system. It is responsible for collecting and interpreting the vehicles surroundings using its sensor data [5]. However, traditional perception systems are often limited to recognizing a predefined set of object categories through methods like 2D/3D object detection, occupancy prediction, and sensor fusion [5]. These traditional methods rely on manual data labeling and annotation which can be slow and ineffective. In contrast, the prediction module is built upon the real-time data from the perception layer to analyze historical trajectories and current status of traffic participants, to predict future behavior [5]. By using textual descriptions, VLMs could provide a richer source of supervision, overcoming the adaptability restrictions of traditional and label-heavy systems [1].

2.3 Deep Learning

Machine learning is a field of artificial intelligence that learns to make predictions or decisions by fitting mathematical models to observed data. They learn patterns from examples, rather than explicitly programming rules for every situation. The goal in this setting is to learn a function that maps an input to an output based on previously observed data [10].

Machine learning methods can be divided into three paradigms: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, a model learns a mapping from inputs to outputs using labeled training examples. In contrast, unsupervised learning learns from the inherent structure of the data itself without any requirement of annotated labels. The goal in unsupervised learning is therefore not to predict a predefined output but rather to understand patterns, clusters, or latent representations in the data [10].

Deep neural networks are a class of machine learning models. They can be described as parametric function approximators that can represent a broad family of relationships between inputs and outputs. They are capable of processing large and complex inputs, including high-dimensional data, and are thus particular useful for unstructured data such as images, videos, and text. They are widely used in modern applications and are considered one of the most practical machine learning models in many domains [10].

An important concept of deep learning is representation learning. This means that neural networks learn internal representations of the data that capture the relevant structure, instead of relying on manually engineered features. These learned representations are often of high dimensions that describe meaningful properties of the input data [10]. For example, when neural networks are trained on large datasets, similar inputs are located close to each other in the learned representation space. This is used in modern multi-modal models, where different modalities such as images, videos, and text are mapped into a shared representation space that makes it possible to compare and retrieve between modalities [6].

2.3.1 Neural Networks and Multilayer Perceptrons

Neural networks are built from creating simple parameterized operations and training them jointly with gradient descent. The Multilayer Perceptron (MLP) is the simplest one. In it, each neuron takes a set of inputs, applies a linear transformation, and then passes it through an activation function [10]. A simplified neuron operation for a linear layer can be written as

$$h = a[\beta_0 + \boldsymbol{\beta}^T \mathbf{x}], \tag{2.1}$$

where $\mathbf{x}$ is the input vector, β_0 is the offset, and $\boldsymbol{\beta}$ is the vector of slopes [10].

Neural networks are built by stacking multiple such transformations in layers. The number of hidden units in each layer is referred to the networks width, while the number of hidden layers as the depth. Together they determines the models capacity, which reflects the flexibility of the network to capture complex relationships in the data [10].

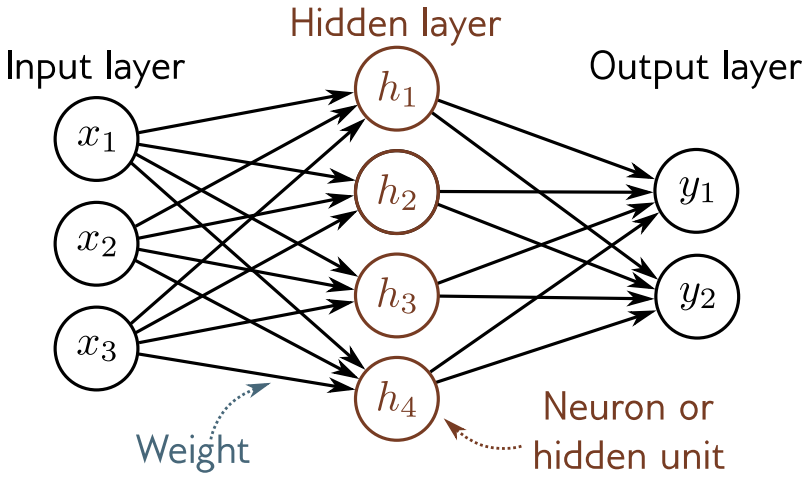


Figure 2.2: Terminology for neural network [10].

A common architecture of neural networks is the Fully Connected Network [10], also referred to as the Multilayer Perceptron (MLP) [11]. These architectures are built from several fully connected layers, meaning that each hidden unit receives input from all units in the previous layer. The goal of an MLP is to approximate complex functions that map inputs to outputs. By stacking multiple layers with nonlinear activation functions, the network can in theory learn any continuous function. In the transformer architecture, MLPs are the core building blocks as they are used as sublayers to to transform representations after the attention operation [10]. An example of a neural network is shown above in Figure 2.2.

2.3.2 Loss Functions and Optimization

In order to train a neural network one must define a learning objective. This objective is specified using a loss function, which measures how far the model's predictions are from the true target. The goal of training is to update the parameters of the

model by minimizing this loss [10]. Given a set of input-output pairs and parameters, the loss can be written as:

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \phi), y_i) \tag{2.2}$$

where $\ell(\cdot)$ measures the error between the predicted and target values

and the optimal parameters are obtained by solving

$$\phi^* = \arg\min_{\phi} \mathcal{L}(\phi) \tag{2.3}$$

The choice of loss function shapes the geometry of what the model learns to represent. Common loss functions are the cross-entropy loss for classification problems and mean squared error in regression problems [10].

For training neural networks one often use gradient descent. This method is based computing the gradients of the loss with respect of the models parameters, and then moving in the step that minimizes the gradients the most for each step. In practice this is done by backpropagation [10]. The equation below show how the parameters get updated at each iteration by moving them in the direction, i.e. the gradient, that reduces the loss:

$$\phi_{t+1} = \phi_t - \alpha \nabla_\phi \mathcal{L}(\phi_t), \tag{2.4}$$

where α is the learning rate.

In practice, neural networks are often trained with Stochastic Gradient Descent (SGD). Instead of computing the gradients on the full dataset it only uses a small random subset of the dataset for each step. This introduces some noise which can help the model to escape local minimias. It also leads to models that are less computationally expensive and that generalize well. Another widely used optimizer is Adaptive Moment Estimation (Adam). Adam adapts the learning rate by adding a momentum term based on the estimate of the gradients and the squared gradient. In practice, Adam often tend to converge faster and are less sensitive of the choice of learning rate compared to gradient descent [10].

2.4 Transformers and Attention Mechanisms

Transformers are the foundation of many modern deep learning systems. This section first explains the transformer architecture and the attention mechanism before introducing the perceiver architecture.

2.4.1 Transformer Architecture

The transformer architecture is a form of neural network architecture that was introduced for sequence modeling [12]. Unlike previous models such as Recurrent Neural Networks or Convolutional Neural Networks that use recurrence or convolution operations, transformers instead use the attention mechanism. This allows the model to handle all elements in a sequence in parallel rather than one by one. This makes the model more efficient and better to capture long-range dependencies [12]. This property is very important for perception tasks, such as modeling how vehicles interact with each other. These objects are rarely independent in traffic scenarios, and by capturing these complex relationships along the sequence, transformers gives a better understanding of the scene.

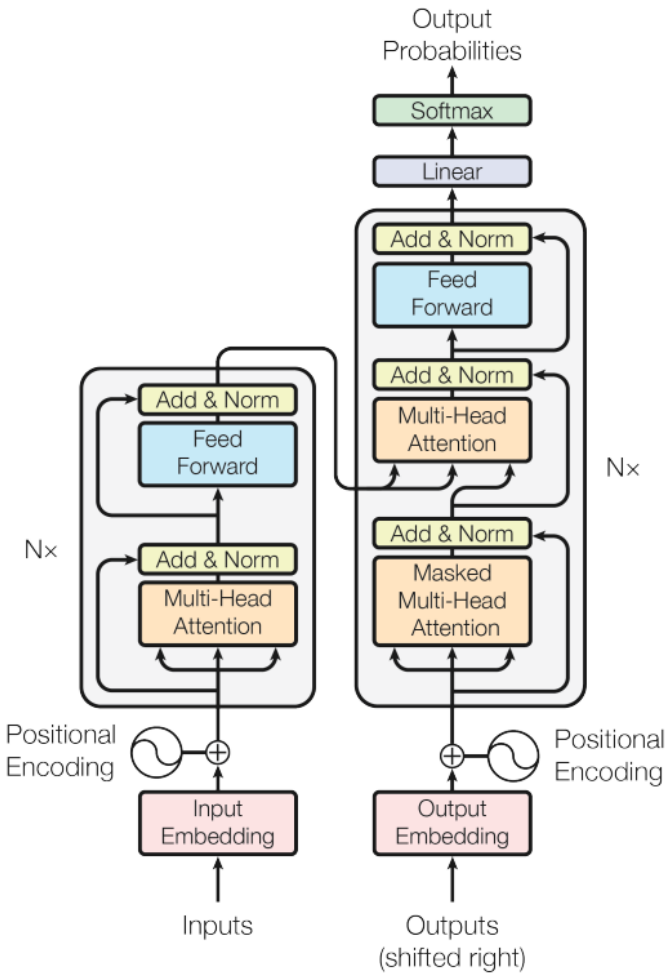


Figure 2.3: The Transformer architecture [12].

A single transformer block consists of several components. It is built of a multi-head self-attention sub layer followed by a feedforward MLP, with residual connections and layer normalization applied after each block [10]. These blocks are then stacked on each other to create the full transformers model. The input to a transformer is first tokenized and then mapped to embeddings. These embeddings are then passed through the stack of transformers layers, where each layer will update the

representation of every token. This is achieved by the self-attention mechanism [12]. The transformer architecture is shown in Figure 2.3.

2.4.2 Attention Mechanism

The core operation inside a transformer is the attention mechanism. An attention function maps a query and a set of key-value pairs to an output vector. The output is calculated by a weighted sum of the values, where the weight assigned to each value depends on the similarity of the query and the keys [12].

Query, keys, and values have the following roles: the query represent what information we look for, the key what information each token contains, and value what information that is passed forward. The similarity calculated between a query and a key defines how much attention the value should have. In practice this is done by the scaled dot-product attention function [12]. It can be written as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.5}$$

where Q, K, and V represent the matrices of queries, keys, and values, and d_k is the dimensionality of the key vectors.

In practice, often several variants of attention are used. One such variant is the multi-head attention, shown in Figure 2.4, where the transformer applies multiple attention operations at the same time. In multi-head attention each attention head learns a different projection of the queries, keys, and values, and all heads are then combined in the output to form the final representation. This allows the model to focus on different relationships in the data [10]. Another type of attention is cross-attention, where the information from two different sequences are combined by letting the queries come from one representation while the keys and values are derived from another [13].

Attention lets every part of the input interact to every other part directly. This makes it easier for the model to learn long-range dependencies such as interactions between different objects, time steps, spatial regions, etcetera [10]. It explains why transformers have becomes such a central architecture in modern perception systems.

2.4.3 Perceiver Architecture

The Perceiver model is built on the transformer architecture and is designed to handle high-dimensional inputs such as images and videos and other type of multi-modal data. Transformers are very flexible but scales very poorly as the input size increases. This because self-attention requires comparing all inputs with each other, which leads to quadratic memory and computational cost. Applying transformers to high resolution systems is thus challenging [14].

To handle this, the Perceiver architecture was introduced. It was designed with a mechanism that decouples the input size from the computational complexity, mak-

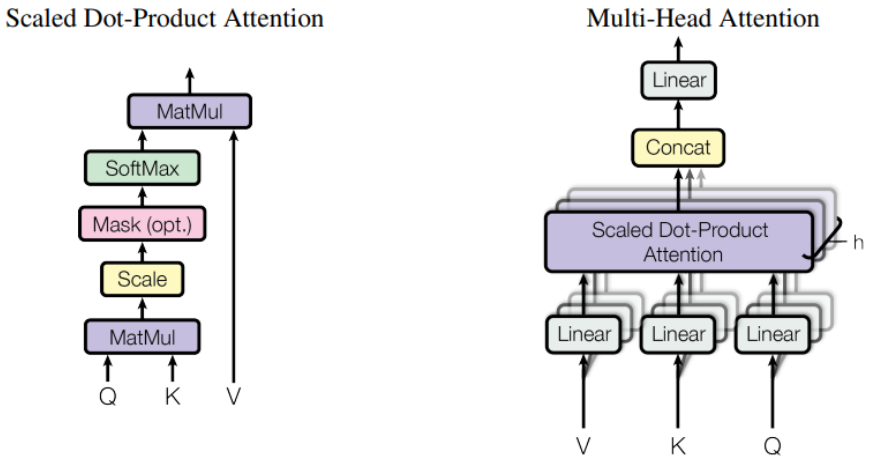


Figure 2.4: Scaled dot-product attention block and Multi-head attention block [12].

ing it scale linear with sequence length rather than quadratically as standard self-attention does. This is what enables the model to handle very high-dimensional inputs. The main idea of the Perceiver was to use a small set of latent variables that act as a bottleneck. The input is projected onto this latent space using cross-attention, which is then processed using transformer self-attention blocks. By refining this representation iteratively, the model can can focus on the most relevant information instead one doing full pairwise attention. This is what decreases the computational cost [14]. An overview of the Perceiver architecture is shown below in Figure 2.5.

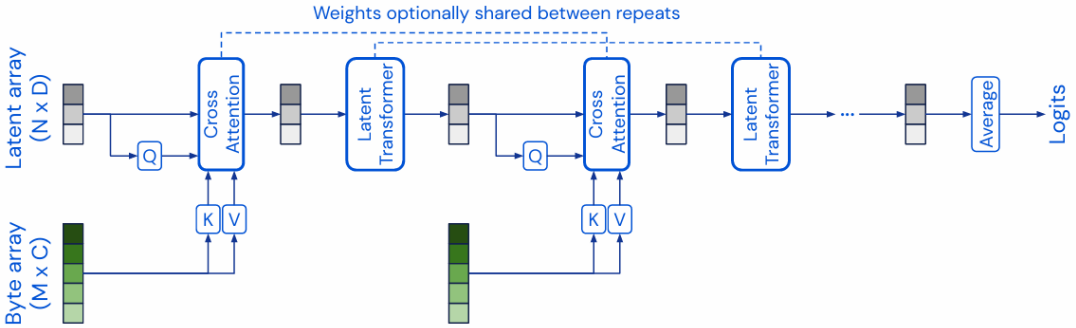


Figure 2.5: Overview of the Perceiver architecture [14]

2.5 Contrastive Learning

Contrastive learning is a paradigm within representation learning and is a central components in many domains, including computer visions and natural language process. The goal is to map positive samples (similar samples) close to each other in the embedding space while at the same time push negative samples (dissimilar samples) far apart. It was first developed in the uni-modal setting where similar samples are the same picture augmented and dissimilar samples are other images in the batch,

as in SimCLR [15]. The same principle generalizes to multi-modality where matching image-text pairs are treated as positive examples while non-matching pairs are treated as negative examples [6].

Within the multi-modal setting contrastive learning is often based on the dual encoder architecture. It consists of one image encoder and a text encoder that maps input to a common embedding space independently. The goal during training is to push matching image-text pairs close to each other and non-matching pairs far apart. Similarity between embeddings is calculated using cosine similarity and a cross-entropy loss over the similarity matrix is optimized during training [6].

After training, the learned embedding has a useful geometric structure. Semantically closed concepts tend to group close to each other regardless of the modality. This makes it possible for retrieving matching images for a query, finding nearest neighbor in the embedding space, and zero-shot classification [16].

2.5.1 Contrastive Vision-Language Embedding Models

The most prominent example of contrastive vision-language embedding model is CLIP. CLIP was released in 2021 and was trained on hundreds of millions of image-caption pairs collected from the internet [6]. Given the large amount of data it is trained on, CLIP has shown to generalize well across a wide range of visual tasks. Another key property of CLIP is its ability to perform zero-shot classification. This means that class labels are replaced by textual prompts and predictions are based on similarity in the embedding space. The advantage of this is that it makes the model more flexible and applicable as it can be used for new tasks without additional training data [17].

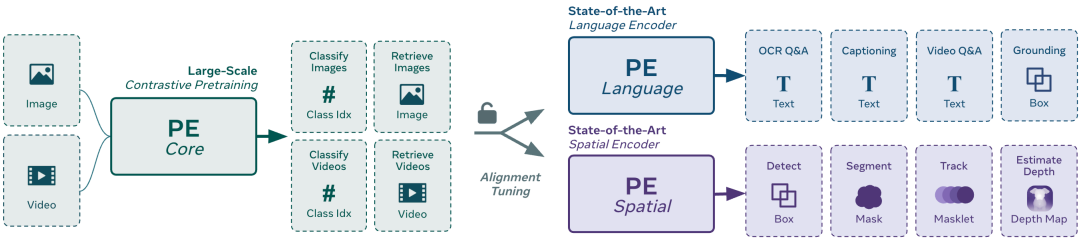


Figure 2.6: Overview of the Perception Encoder (PE) [7]

More recent work introduced the Perception Encoder (PE), shown in Figure 2.6. As CLIP, it is based on a vision encoder and a text encoder trained with a contrastive learning objective. CLIP's training objective is to capture overall semantic meaning, and important spatial and temporal details are therefore often disregarded in the learned representations. PE addresses this by showing that these features actually exists, but that they are hidden in intermediate layers of the network and not in the final layer [7].

To expose and preserve these features in the final representation, PE introduces two types of alignment methods: language alignment and spatial alignment. Language alignment is designed to make the final embedding more compatible for language

models, meaning that the visual representation can be better understood and used by the language model. This is important for tasks such as captioning, where the model need to be able to understand and reason over visual and textual information. The second alignment method, spatial alignment, was designed to extract and retained spatial information required for tasks such as detection, depth, tracking and segmentation. By introducing these two alignment methods, PE has achieved better performance than CLIP [7]. Another key feature of PE is that it can handle both images and videos, unlike CLIP which only handle images [6]. Videos are handled by encoding individual frames and thereafter averaging their embeddings [7].

2.5.2 Captioning Engines in ADAS and AD

Captioning engines are central for autonomous driving. Their aim is to general a textual description for a given scene, such as an image, a point cloud, or an object within in the scene, for example interactions between traffic participants and the structure of the road [5]. Many studies have treated captioning as a special case of visual question answering, where a VLM is used to answer a fixed set of questions. However, this approach can be challenging as several studies have found that VLMs can fabricate answers to driving-related questions when visual information is absent [9]. VLMs also tend to demonstrate a limited capacity to identify visual corruption when it is not explicitly prompted. In the absence of those specific prompts, VLMs tend to overlook or misinterpret corrupted visual features [9].

To evaluate captioning and related retrieval tasks some methods that are commonly used are Top-N accuracy or Recall@K metrics. These are used to measure the alignment between the visual inputs and textual outputs [5]. Top N-accuracy calculates the ratio of correctly predicted questions to the total number of questions, while Recall@K measures the similarity among the K most similar predicted captions based on an embedding similarity measure [5].

Rule-based captioning can be used to complement approaches based on VLMs. While VLMs are effective at understanding visuals at a high level, they often lack spatial grounding and precise reasoning capabilities [3]. This motivates the use of rule-base methods in combination with VLMs [2]. Rule-based captioning relies on sensor measurements and annotations to generate deterministic descriptions of objects, distances, lane positions, and motion patterns. This could for example be attributes such as speed, acceleration, and trajectory curve [18]. However, rule-based captioning lack the flexibility and richness and may overlook higher level of semantic details. Therefore, is rule-based captioning most effective when used in combination with captioning from VLMs [18].

In addition, a known challenge for VLMs is hallucination, where the model tends to produce descriptions not grounded in the visual content. To mitigate this issue, some studies have shown that the captioning engine can first generate rule-based captions and thereafter provide them as contextual input to the VLM, together with a prompt instructing it to supplement the captions with additional information not covered by the rules, for example weather conditions, lane structure, and other relevant attributes of the scene [18]. Hence, providing rule-based captions can in

theory reduce the hallunications, improve the reliability of the captions and thus increase the quality of the output for downstream tasks [18].

3

Methods

This section outlines the scientific approaches employed to address the thesis challenges: developing a captioning engine and validating its effectiveness through fine-tuning a contrastive vision-language embedding model. The approach has three parts: preparing the data, constructing the captioning engine through a staged ablation, and the fine-tuning using the generated captions.

3.1 Dataset

The thesis uses the Argoverse 2 dataset [19]. It is a collection of driving data recorded across six U.S. cities: Austin, Detroit, Miami, Pittsburgh, Palo Alto, and Washington. The dataset contains driving scenarios of different seasons, weather conditions, and times of day, to ensure a diverse set of traffic environments [19].

Argoverse 2 consisted of four datasets: the sensor dataset, the motion forecasting dataset, the LiDAR dataset, and the map change dataset. The focus in this thesis was on the sensor dataset. The sensor dataset contained 850 fully annotated sequences and 150 single-frame annotated sequences. The fully annotated sequences spanned 15 seconds and were sampled at 10 Hz, providing roughly 150 frames per sequence. The dataset was divided into 700 fully annotated training, 150 validation as well as 150 single-frame annotated test sequences [19].

The ego vehicle and dynamic agents such as cars, pedestrians, buses and motorcycles were fully annotated with 3D cuboids in each sequence. Each sequence contained on average 75 annotated agents, where each agent included positional coordinates relative to the ego vehicles position. The dataset also contained a detailed map for each sequence, which included positions of lanes, lane boundaries, drivable area, and intersections [19].

To give a sense of the data used for this thesis, Figure 3.1 shows the map and the ego vehicles trajectory for a specific sequence.

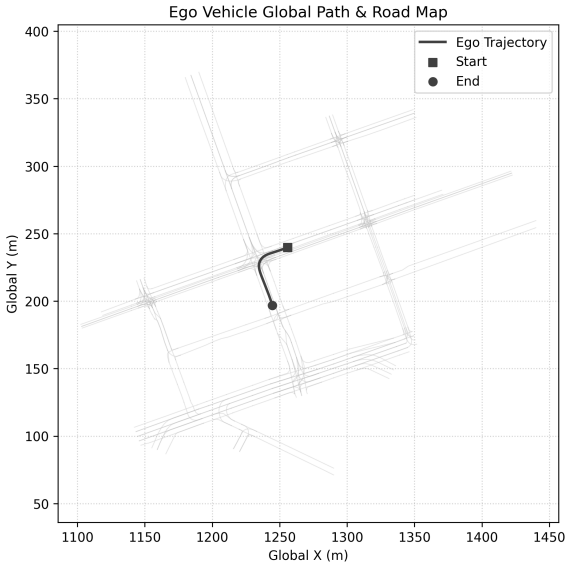


Figure 3.1: Plot of a sequence. Showing the map and the ego vehicle's trajectory.

3.2 Data Preprocessing

Before the event detection could proceed, the data requires several preprocessing steps. These aimed to transform the coordinates of the agents and remove non-relevant agents in order to improve the evaluation of the rules and enable faster computation.

Coordinate transformation

The most important step of the data preprocessing is transforming the coordinates of the agents. The ego vehicle and the map are defined in world coordinates while all agent positions are relative to the ego's position. Without this transformation, it would not have been possible to perform any calculations involving the positions of the ego vehicle and the agents.

For an agent at ego-relative position $(x_{\text{rel}}, y_{\text{rel}})$, given the ego vehicle's world position $(x_{\text{ego}}, y_{\text{ego}})$ and heading ψ, the world position is:

$$\begin{bmatrix} x_{\text{world}} \\ y_{\text{world}} \end{bmatrix} = \begin{bmatrix} x_{\text{ego}} \\ y_{\text{ego}} \end{bmatrix} + \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x_{\text{rel}} \\ y_{\text{rel}} \end{bmatrix}. \tag{3.1}$$

Agent filtering

Not all tracked agents were relevant to the maneuver classes targeted by our rule-based system. Pedestrians, cyclists, and stationary objects were excluded, as were agents traveling in opposing or largely perpendicular directions. Agents were excluded based on their heading alignment compared to the ego's. Cosine similarity between agents' headings was compared to ego's at timestamp t. $\mathcal{A}$ denoted the set of candidate agents in a sequence and $\mathcal{T}_i$ the set of timestamps at which agent

$i \in \mathcal{A}$ and the ego vehicle were co-observed. At each $t \in \mathcal{T}_i$, heading alignment was computed as

$$\alpha_i^{(t)} = \frac{\mathbf{v}_i^{(t)} \cdot \mathbf{v}_{\text{ego}}^{(t)}}{\max\Big(\|\mathbf{v}_i^{(t)}\|,\, \epsilon\Big) \cdot \max\Big(\|\mathbf{v}_{\text{ego}}^{(t)}\|,\, \epsilon\Big)} \tag{3.2}$$

where $\mathbf{v}_i^{(t)}$, $\mathbf{v}_{\text{ego}}^{(t)} \in \mathbb{R}^2$ were the world-frame velocity vectors of agent i and the ego vehicle, and $\epsilon = 10^{-6}$ avoided division by zero. If $\alpha_i^{(t)} \geq 0.5$, which corresponded to a maximum heading deviation of $\arccos(0.5) = 60°$, exceeds the threshold the agent was discarded.

3.3 Captioning Engine

The captioning engine was central to this method. It was designed as a staged ablation: each stage was built on the previous stage and was evaluated independently. In this work, we included (i) rule-based captions derived from map and trajectory data, and (ii) video-level captions that incorporated temporal context.

3.3.1 Rule-Based Captions

The first stage of the captioning engine was a deterministic system for detecting cut-ins, cut-outs, and ego lane changes, based on calculations involving the trajectories of the ego vehicle and other agents, as well as the geometry of the map. These are events that VLMs struggled to identify on their own. The idea was to detect these events through explicit calculations and thereafter provide them as contextual input to the VLM in the later stages of the captioning engine.

Building lane centerlines

The map in Argoverse 2 represented roads as a graph of lane segments. Each lane segment stored its left and right boundary positions and references to its predecessor and successor segments. The lane segment's centerline was not stored, instead it was approximated by taking the midpoint between the left and right boundaries at each point along the segment.

To build a forward-looking centerline for the ego vehicle, we constructed a system that for every second, s, sampled the ego position and identified the closet lane segment at that position. It then followed connected lane segments step by step, adding their centerlines to the path until the accumulated length was the desired look-ahead distance. This look-ahead distance could either be a fixed distance, for example 30m, or depend on the ego vehicle's speed, so that we did not build a long centerline if the vehicle isn't moving.

The look-ahead distance is speed-adaptive:

$$d_{\text{look}} = \text{clip}\Big(v_{\text{ego}} \cdot T_{\text{horizon}},\; d_{\min},\; d_{\max}\Big), \tag{3.3}$$

where v_{ego} is the current ego speed, T_{horizon} is a fixed time horizon, and $d_{\min}$ and $d_{\max}$ are lower and upper bounds that prevent the distance from becoming unreasonably short or long.

An important thing to have in consideration when building centerlines was how to handle when a lane segment had multiple successors. If this was the case our system looked $1s$ ahead in the future, and found the segment the ego vehicle was closest to at the future position. This ensured that the system committed to the centerline aligned with the ego vehicle actual path rather than an arbitrary branch.

Cut-in and cut-out detection

The laterally distance $d(t)$ was tracked over time for each agent relative to the ego vehicle. Two thresholds based on current lane width w define three zones:

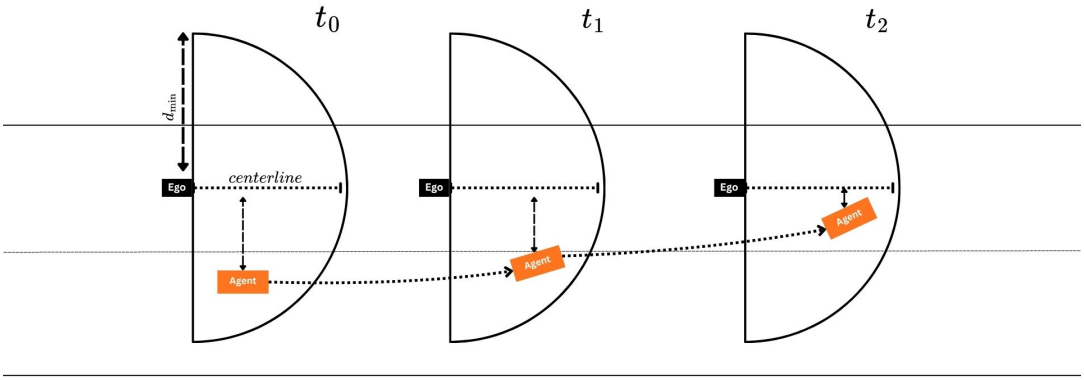
$$\theta_{\text{in}} = 0.50 \times w, \tag{3.4}$$

$$\theta_{\text{out}} = 0.60 \times w. \tag{3.5}$$

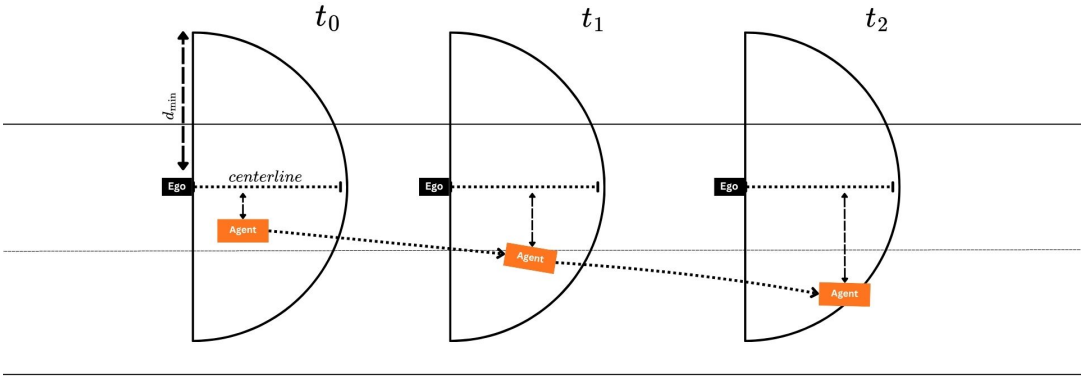
An agent was considered to be inside the ego lane when $d < \theta_{\text{in}}$, and outside when $d > \theta_{\text{out}}$. It was important to include a gap between those two thresholds in order to introduce hysteresis, which stabilizes classification by preventing rapid oscillations when an agent is near the lane boundary. Furthermore, only agents inside a forward half-disk of radius $d_{\min}$ around the ego vehicle.

A cut-in was defined as when an agent transitioned into the ego lane from the outside region and stayed there for at least some consecutive sampled frames. A cut-out was conversely a transition from the ego lane to the outside region under the same conditions. We sampled every 10th timestep, corresponding to a sampling rate of 1 Hz, and set the minimum number of consecutive frames to 10. Thus, each agent was required to remain in the ego lane for at least 10 frames, that is, 1 second, for it to be confirmed as an event. This requirement was important for filtering out noise and projection artifacts.

Figure 3.2 shows examples of a cut-in and a cut-out that are detected by the rule-based classifier. Within the forward half-disk of radius $d_{\min}$ each agent is assigned a projected lateral distance to the ego lane centerline. A cut-in is registered when the projected distance from agent to the ego vehicle centerline drops below the lane-width threshold. A cut-out is the transition in the other direction.



(a) Cut-in: the agent crosses from the adjacent lane toward the ego centerline.



(b) Cut-out: the agent departs the ego lane toward the adjacent lane.

Figure 3.2: Illustration of cut-in and cut-out maneuvers that are detected by the rule-based classifier.

Ego lane change detection

A lane change was detected when the ego vehicle changed lane from its lane to a neighboring segment in the lane graph. For each transition the ego vehicle made, we checked whether that boundary was legally crossable; for example, a dashed white line allowed lane crossing while a solid double yellow did not. This check was important in order to filter out movements that the ego vehicle legally could not have done.

Intersections were particularly tricky as they were built up by many different lane segments and the ego vehicle often drifted across these segments, which made it look like a lane change had taken place. Without special handling, this could have led to many false positives, where lane changes were detected in intersections even though none had occurred. We handled this by implementing a check: whenever a lane change was detected, we looked ahead over a short time window on the ego vehicles trajectory. If the vehicle returned to its initial lane, we assumed the crossing was incidental and the event was discarded.

Another complexity of lane changes involved the splitting and merging of lane segments. A split occurred when a single lane segment led into two or multiple segments, while a merge was when multiple segments converged into one. In both situations, the system computed the heading difference between the available segments and if the heading difference between two candidate segments was below 4.5°, the vehicle was considered to be continuing straight ahead. If the heading difference exceeded that threshold, the segments were diverging or converging at a meaningful angle, and the transition was instead classified as a lateral move.

The threshold of 4.5° was determined empirically by analyzing false positive lane-change detections on sequences with known ground truth and selecting the value that minimized the number of false positives. This rule was important as it allowed the system to distinguish between a lane that widened into multiple lanes and cases where an actual left or right lane change had taken place.

Another difficulty happen after the ego vehicle performed the lane change, as the centerline the ego vehicle referenced shifted with it. As a result, agents that were previously within the ego lane appeared to be outside the ego's lane afterwards, which could incorrectly trigger a cut-out. Vice versa, a cut-in could incorrectly be triggered since agents that were previously outside the ego lane may appeared to be inside it. To prevent this behavior, the tracking timeline was split whenever a ego lane change was detected and all histories, lane states, and event counts were reset so that previous lane geometry did not affect later detections. We also removed 0.5 seconds before and after each lane changes, so no events were detected during the transition.

Parameters of the rule-based method

Table 3.1 lists the final configuration of the parameters. Values was selected by iterative experimentation on manually annotated validation set of 75 samples. Our configuration deliberately favored precision over recall as a false caption is more damaging to downstream training as a missed event was.

Parameter	Value	Description
`CHECK_EVERY_NTH`	10 frames	Sampling interval; effective rate ≈1 Hz at 10 Hz input.
`MIN_DETECTION_FRAMES`	10	Minimum consecutive samples inside/outside to confirm an event (≈1 s).
$d_{\min}$	30 m	Lower bound on look-ahead distance.
$d_{\max}$	150 m	Upper bound on look-ahead distance.
T_{horizon}	4 s	Time horizon for speed-adaptive look-ahead.
θ_{in}	$0.50 \times w$	Lateral threshold for inside-lane classification. w is lane width.
θ_{out}	$0.60 \times w$	Lateral threshold for outside-lane classification. w is lane width.

Table 3.1: Parameter settings for the rule-based detection system.

3.3.2 Video Captions

Using the rule-based detections as input, a VLM was used to generate detailed and contextually rich captions for each sequence. The choice of model was an important consideration. We went for Qwen3-VL due to its strong performance and open-source availability. The model had both an instruct variant that answer directly and a thinking variant that first reasons step-by-step before producing its final answer [20]. We selected the instruct model since we found that the thinking model was prone to entering reasoning loops and hallucinating additional events not present in our rule-based detections. The instruct model, in contrast, better followed the provided input rules and was therefore more reliable.

Prompt engineering was another important design decision for the video-level captions. We considered multiple implementations for it. One approach we considered was to generate three separate captions per sequence. One describing the environment, one the vehicles involved, and one the detected maneuvers. These three independent captions could thereafter be merged into a single caption. However, in practice this proved difficult to evaluate and often produced cluttered captions. The merged captions devoted most of their length to environmental descriptions and only a small portion to the maneuvers themselves. This is likely because environments offer more visual detail to describe such as buildings, road markings, weather, and traffic. The maneuver content was often diluted and generic in the final caption, which is the opposite of what we wanted for the retrieval task.

Instead of the above approach, we designed three captioning templates of increasing complexity for prompting the VLM: a minimal, an intermediate, and a complex template. The minimal template focused solely on what occurred in the sequence

based on the rule-based detections, with little environmental description. The intermediate template extended this by including descriptions of the vehicles involved, temporal progression, and relevant traffic context. The complex template further added descriptions of the environment, road conditions, weather, and infrastructure.

To reduce hallucinations and produce more grounded captions, we enriched the VLM prompt with metadata derived from the Argoverses' annotations. For each sequence containing a cut-in, cut-out, or lane-change, three more fields of information are injected into the prompt (Figure 3.3). First, the vehicle category, this includes sedan, truck, bus or SUV's. This helps the model connect each maneuver to the correct agent. Second, the distance from the ego-vehicle to the agent performing the maneuver. Third, a unique agent identifier. When multiple vehicles perform events within the same sequence, each is assigned an unique label connected to that specific agent (A, B, ...), allowing the caption to refer to them uniquely rather than confusing and mixing the actions of different agents.

This approach allowed us to generate training data while maintaining consistent grounding in the ground truth of our rule-based detections.

Complex Captioning Template

System prompt:
You are an expert autonomous driving scene captioner. You are given a sequence of dashcam frames and a list of verified maneuver detections. Write a caption that covers all of the categories below.
Terminology: The *camera vehicle* is the vehicle the dashcam is mounted on and is not visible in the frames — all vehicles visible in the frames are other traffic participants. A *cut-in* means another vehicle enters the camera vehicle's lane from an adjacent lane; a *cut-out* means another vehicle leaves the camera vehicle's lane by moving to an adjacent lane; a *lane change* means the camera vehicle itself changes lanes.
Most critical rule: The detection list is complete.

- If the list is empty, the words *cut-in*, *cut-out*, *lane change*, or any synonym must not appear in the caption, even if vehicles appear to move between lanes.
- If the list contains maneuvers, mention each exactly once and invent no additional maneuvers.
- For empty lists, describe traffic neutrally.

Maneuver present: Keep it consistent. A cut-in agent must be visibly entering the ego lane. A cut-out agent must be visibly leaving it. Multiple listed maneuvers may involve different vehicles. If no specific agent can be confidently identified, describe the maneuver generically rather than guess.
What to include:

1. **Maneuvers:** mention every listed maneuver exactly once using the exact term provided. Other behaviors (braking, yielding, stopping, accelerating, passing) may be described.
2. **Vehicles:** restricted body-type vocabulary (sedan, SUV, pickup truck, van, bus, truck, motorcycle, bicycle). Avoid terms like "tracked" or "military" unless clearly justified. Include color and relative position.
3. **Temporal progression:** preserve the order of listed maneuvers and keep each attributed to a single agent.
4. **Traffic context:** surrounding density, pedestrians, cyclists, parked vehicles.
5. **Environment:** road type, number of lanes, surface condition, lane markings.
6. **Conditions:** weather, lighting, time of day, visibility.
7. **Infrastructure:** intersections, ramps, signs, construction zones, barriers, guardrails, traffic signals.

Grounding rules: describe only what is directly visible. Do not invent highway numbers, route numbers, street names, business names, or city names. Omit a maneuver's direction if not clearly visible.
Format: a cohesive paragraph of 6–8 sentences; no bullet points, headers, or numbered lists. No repetition of the detection list.

User prompt:
Detected maneuvers: `{maneuver_str}`. Write a caption of the driving sequence following the instructions in the system prompt. If the detection list is empty, describe the scene without using any maneuver terms. If maneuvers are listed, attribute each to a single specific vehicle that is visibly performing that action.

Figure 3.3: The complex captioning template used to prompt Qwen3-VL-Instruct.

3.4 Fine-Tuning of CLIP and PE

This section describes the second central part of the method, where a contrastive vision-language embedding model is finetuned. It covers the model configuration and the fine-tuning strategy used.

3.4.1 Model Configuration

A core design choice was which model to use for our purpose. Ultimately, four different configurations were used in order to identify the impact of backbone selection and fine-tuning. As a baseline an out-of-the-box ablation of CLIP ViT-L/14 was used as our zero-shot baseline. Then as complement a fine-tuned CLIP ViT-L/14 highlighted the difference our fine-tuning strategy had on performance. Similarly, a baseline and fine-tuned version of Meta's PE-Core-L14-336 was used as models of interest in order to measure the impact of more modern architecture. By comparing all four, we could attribute performance differences to either fine-tuning or backbone design.

3.4.2 Fine-Tuning Strategy

The captions produced by the captioning engine were used to fine-tune a pretrained CLIP and PE model via contrastive learning. The encoded frames of a dash-cam sequences were averaged in order to represent a sequence as a single embedding.

Freezing strategy

A common design principle when fine-tuning is to freeze early layers of the transformer blocks and only fine-tune later layers. In our case, we froze the eight first layers of the transformer blocks (0–7 of 12), while only the later layers (8–12) were fine-tuned. The early layers capture low-level features such as edges, colors and textures, which are largely universal and do not require adaption. In contrast, later layers encode higher-level semantics and must adapt to the the target domain. This was important in our setting, as dash-cam data differs a lot from typical pretrained images: vehicles are often observed behind and partially obscured by other road agents, rather than appearing in clean perspectives.

It was also necessary to freeze the positional embeddings for both the text and visual encoders. As these embeddings contains structural information learned from the pretraining phase, updating them could degrade the information that was learned. Similarly, class embeddings and token embeddings were also frozen. In summary, the model's infrastructure and structural components were frozen in order to only fine-tuning the feature representation.

Sliding window sub-sequences

The 700 training sequences provided a limited number of distinct scenarios for fine-tuning a large encoder. To address this and expand the dataset, each sequence was

split into overlapping sub-sequences using a sliding window. For example, a sequence of 15 seconds with a 10-frame window and stride of 1 produced 6 sub-sequences. Each sequence captured a slightly different temporal context and therefore yielded a slightly different caption. By using this strategy the number of video-caption pairs was increased substantially without the need to collect additional data.

Data augmentation

To reduce overfitting on repeated sub-sequences, a standard data augmentation pipeline was used. It included random horizontal flips, color jittering, and cropping. Horizontal flips is when the image is mirrored along the vertical axis. The corresponding captions were also updated in this steps, so that left-to-right lane change became a right-to-left lane change. Color jittering was used in order to make the model more robust in variations in lighting conditions. Cropping was used to standardize the dimensions of the images, but also to reduce barrel distortion that are often found in the edges of wide-angle lenses.

The data augmentation part was also important for making the sub-sequences produced by the sliding window more diverse. Argoverse2 data was shot in good lighting, augmentation lets us generate other conditions, such as darker scenes that would represent driving in evening / night.

Pooling of frames

Another key design decision was how to represent a sequence of frames as a single embedding. Our implementation followed the approach used in the Perception Encoder paper [7]. Four frames was sampled uniformly from each sequence and then embedded individually. The resulting frame-level embeddings were then aggregated using mean pooling to produce a single embedding per sequence. The idea by averaging embeddings was to capture the the main representation occurring across the sequence.

3.5 Evaluation

This section outlines the evaluation step of the method. It describes evaluation of the rule-based classifier, caption generation, and the fine-tuning of the contrastive vision-language embedding model.

3.5.1 Rule-Based Classifier

The rule-based classifier produces deterministic event labels from trajectory and map data, which makes it possible to compare the labels to a ground truth label. Evaluation was therefore framed as a per-event classification problem over the three targets: cut-ins, cut-outs and lane changes.

A held out set of 75 annotated sequences from the Argoverse2 dataset was manually annotated. For each sequence, the annotator recorded which of the three target

events were present. Sequences without any events were labeled as such.

For each type of event, precision, recall and F1-score was computed along with the count of true positives and negatives. The counts along with the metrics produced a way to make distinctions about what the model was better or worse at classifying. This distinction matters since false positives teaches the downstream encoder to associate a maneuver with frames that does not include a maneuver.

The distribution of detected events across the dataset was also examined as part of the evaluation. The manual annotator and the rule-based classifier operate under different assumptions about which events are relevant. The classifier applies a distance filter through the look-ahead parameter $d_{\min}$, while the annotator does not. As a consequence, an event judged relevant by the annotator may legitimately fall outside the range of the system, producing what appears as a false negative but is in fact a discrepancy in scope rather than a detection failure. Comparing the event distributions produced by the two procedures provided a way to examine whether the classifier was over- or under-firing at a given hyperparameter setting.

3.5.2 Captioning Engine

The choice of model was evaluated. Different models hold different number of parameters. Some models has thinking properties, allowing the model to go more in depth in the task given to it. The scale of the model was evaluated by comparing captions produced by the different sizes. Captions were generated with both Qwen3-VL-2B and Qwen3-VL-30B-A3B on the same sequences, and the outputs were compared on grounding, hallucination rate, and the level of detail recovered from the frames.

When a sequence contained events detected by the rule-based stage, the prompt was enriched with structured information about each event: the event type, the agent involved, and the temporal ordering. For example, a sequence containing both a cut-in and a subsequent cut-out by the same vehicle would explicitly state that "Vehicle A performs a cut-in, then the same Vehicle A performs a cut-out". This moves agent tracking and temporal reasoning from Qwen to being handled by the rule-based stage. The effect of infering this additional information in the prompt at generation time was evaluated by studying captions produced with and without the additional context enrichment. Additionally, the extra context was also studied by comparing the results of the downstream fine-tuned contrastive vision-language embedding model.

After producing a caption, the quality of the caption was evaluated using human input. Caption quality was manually assessed along two aspects that are of importance to the downstream retrival task. First, *manuever descriptions* - weather the caption describe the maneuver correctly. Given that the model is enforced to include these descriptions in inference the quality of the description matters more than its inclusion. Second, *scene description*, this includes descriptions of vehicles, temporal progression, traffic, environment, conditions and infrastructure. Judging whether the caption matches the frames or adds details the model cannot actually see. This is important as our complex template (Figure 3.3) enforce the VLM to include de-

tails about environment descriptions. Both said properties matter for fine-tuning. A caption that misdescribes the maneuver or hallucinates environmental descriptions teaches downstream fine-tuned model the incorrect sequence-caption associations.

3.5.3 Fine-Tuning of CLIP and PE

Evaluation was necessary when fine-tuning the contrastive vision-language embedding model. Multiple method was used to better understand the model's performance, since different metrics capture different aspects of the models strenghts and weaknesses.

Two model configurations were evaluated in parallel: CLIP (ViT-L/14) and the Perception Encoder (PE-Core-L14-336). For each configuration, a zero-shot baseline was established by evaluating the pretrained model on the retrieval task without any fine-tuning. The fine-tuned model was then evaluated under the same hyperparameter configurations, and improvements were reported relative to its own baseline.

Recall@K

Recall@K is a common method of evaluating a fine-tuned contrastive vision-language embedding model using querying. It measures how often the correct match appear in the Top K results of a ranked retrival list. Given a query, for example a caption from our validation set, the model ranks all videos by cosine similarity. If the correct video appears in the top $k \leq K$ results it counts as a hit. Recall@K is defined as the fraction of those queries that resulted in a hit in the top K results.

$$\text{Recall@K} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\text{rank}_i \leq K] \tag{3.6}$$

where N is the number of queries and rank_i is the position of the correct match for query i in the ranked list.

For the video-to-text direction, the retrieval is reversed: given a pooled video embedding, the model ranks all candidate captions by cosine similarity and the metric measures how often the correct caption appears within the top K results.

Recall@K was evaluated for $K \in \{1, 5, 10\}$. R@1 measures how often the model retrieves the exact correct match as its top result, while R@5 and R@10 measure how often the correct match appears within the top 5 and top 10 results, respectively.

Median Rank

A complement to Recall@K is the median rank of the correct match across all queries. It measures the median position of the correct match across all ranked retrieval lists. Thus, using both the median and mean rank further nuanced the results, as a large gap between the two indicates the presence of outlier queries where retrieval fails significantly. As with Recall@K, median rank is computed for both video-to-text and text-to-video queries.

Alignment

Alignment measure how similar embeddings of matched pairs are on average. For a set of N matched video-caption pairs, it is computed as the mean cosine similarity between each correct pair's embeddings:

$$\text{Alignment} = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{v}_i \cdot \mathbf{t}_i}{\|\mathbf{v}_i\| \|\mathbf{t}_i\|} \tag{3.7}$$

where $\mathbf{v}_i$ and $\mathbf{t}_i$ are the video and text embeddings of pair i.

Since contrastive learning aims to embed matching pairs similarly to each-other in embedding space, a high alignment-score signals that the model successfully pushes these pairs closer to each other. In our case, a low alignment score would mean that our captions were not enough for the model to distinguish between the caption-video pairs. This could indicate that the captions were either too sparse or too noisy, or that the model had not been fine-tuned enough.

Uniformity

Uniformity was also used to complement alignment, as alignment measure how spread out non-matching pairs are to each-other. In many cases, a model could collapse all embeddings into a single point in the representation space. In that situation the alignment would improve since matching pairs become close. The model is however not learning anything since the representation is not meaningful as all inputs are mapped to almost the same identical embedding. Uniformity punishes this collapse by measuring the average pairwise Gaussian kernel between every embedding:

$$\text{Uniformity} = \log \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} e^{-2\|\mathbf{z}_i - \mathbf{z}_j\|^2} \tag{3.8}$$

where $\mathbf{z}_i$ are the L2-normalized embeddings and the sum runs over all pairs including $i = j$.

A more negative value indicates better uniformity. Together with alignment, uniformity gives a complete picture of embedding representation quality.

Event breakdown

A distinction had also have to be made for the three different event-types included in our thesis objective, since the model could be performing differently with different types of events. For example cut-ins and cut-outs could have great Recall@K while the model collapses and performs worse when it comes to lane-changes. To mitigate this, Recall@K and median rank was evaluated by event-type.

3.6 Method Discussions

This section goes over the main limitations of the method and discuss relevant ethical and risk-related considerations for its use in autonomous driving application.

3.6.1 Limitations

The work was subject to several limitations related to the the rule-based detection pipeline, the generating of captions, and fine-tuning of the encoder.

The rule-based system was restricted to the pre-defined maneuvers: lane-changes, cut-ins, and cut-outs. It did not capture other maneuvers such as the ego vehicle turning left or right or agents braking or accelerating. As a result, the generated captions only reflected a subset of all maneuvers, which limited the diversity of training data.

Another limitation was the dataset itself. The rule-based detection was based on calculating heading alignment and lateral distance, using thresholds to determine if any maneuver had taken place. If there were any errors in the used dataset - for example, if lane positions were not accurate, it would directly affect the outcome. Furthermore, as the rule-based detection was used as the ground truth, any misclassified maneuvers propagated to the captions; the captions would therefore also be incorrect. Another limitation of the dataset was that it was restricted to six cities in the U.S., which could limit the generalization.

The dependence of the generating of captions by the VLM to the rule-based system was another limitation. If any event was misclassified or missed by the rule-based system, it was directly reflected in the generated captions. Hence, the quality of the captions was dependent on the quality of the rule-based system. It was therefore important to design a rule-based system that favored precision over recall, as we rather had identified maneuvers that were correct with some maneuvers missed than many false positives maneuvers.

The generation of the captions also introduced some noise. The VLM hallucinated sometimes and did not provide a rich, detailed caption. An example is that the VLM described the color of the car making a cut-in incorrectly. This weakened the alignment between the video and the caption which could have negatively affected the contrastive learning during fine-tuning. It was therefore necessary to set up strict guidelines for the VLM so that the model would give accurate descriptions.

Finally, the evaluation part of the method was another limitation. The rule-based system was evaluated on a small subset of data that was manually annotated. A larger validation set might have be needed to capture more edge cases. As these annotations were based on human judgment, they limit reproducibility. In addition, using Recall@K and median rank for evaluating the fine-tuning did not measure how good the representation is. Understanding the semantic correctness in downstream tasks remained difficult.

3.6.2 Risk Analysis and Ethical Considerations

Several risks and ethical considerations are associated with the use of VLM-based captioning systems for autonomous driving data. One primary risk is that VLMs can generate captions that are plausible but not grounded in the actual truth. For example, previous studies have shown that VLMs can fabricate and generate answers that look like high quality, even though visual information is heavily degraded or entirely absent [9]. In the context of this thesis, incorrect or misleading captions of driving scenarios may negatively affect the downstream analysis and lead to incorrect conclusions in dataset curation or model development if those captions are treated as reliable descriptions of the scenarios.

A closely related challenge is the limited ability of VLM's to evaluate the reliability of their generated captions. While VLMs can exhibit some awareness of visual corruption, they often only acknowledge these problems when they are explicit prompted for it and otherwise tend to generate confident captions regardless of the quality of the input data [9]. This raises concerns regarding the transparency and trustworthiness of the use of these technologies and it is therefore important to clearly document the limitations of the generated captions.

There also exist broader risks with the increased use of VLMs and ADAS technologies. For example, the possibility of manipulation of the system or incorrect decision making due to compromised inputs or models [5]. In addition, ADAS technologies may negatively affect the driver behavior by diverting attention from the road or reducing trust in the system as a result of frequent or unnecessary warning systems [8]. Although these aspects are not use cases of the system developed in this thesis, it is important to have these considerations in mind for broader development of ADAS systems based on VLMs.

From an ethical perspective, off-the-shelf VLMs trained on large-scale web datasets may reflect biases present in their training distributions, which raises general questions regarding fairness, accountability, and trustworthiness when such models are applied without further adaptation [5]. A related and well-documented limitation of general-purpose VLMs is hallucination, where the model tends to produce output not fully grounded in the visual content [18]. Adapting foundation models to the target domain through techniques such as contrastive fine-tuning is one direction explored in the literature to mitigate these limitations [5], [18], and improving the grounding and reliability of VLMs for driving-scene understanding is one of the motivations for the approach developed in this work.

Potential approaches to improving the safety and reliability of a VLM-based system is to include a step of fine tuning and alignment. An example is to use reinforcement learning from human feedback and supervised alignment tuning. These methods aim to better control the behavior and strategy of the model in order to ensure that the models are consistent with safe and responsible driving principles [5]

4

Results

In the following chapter, the results connected to our objectives are presented. Each section highlights the comparison between a baseline and our implementation.

4.1 Rule-Based Classifier

This section presents the results on using the rule-based classifier on the annotated sequences and its behavior when applied on the non-annotated dataset.

4.1.1 Evaluation on Annotated Data

An evaluation is performed on the 75 annotated sequences. The distribution of the event types in the annotated data is seen in Figure 4.1.

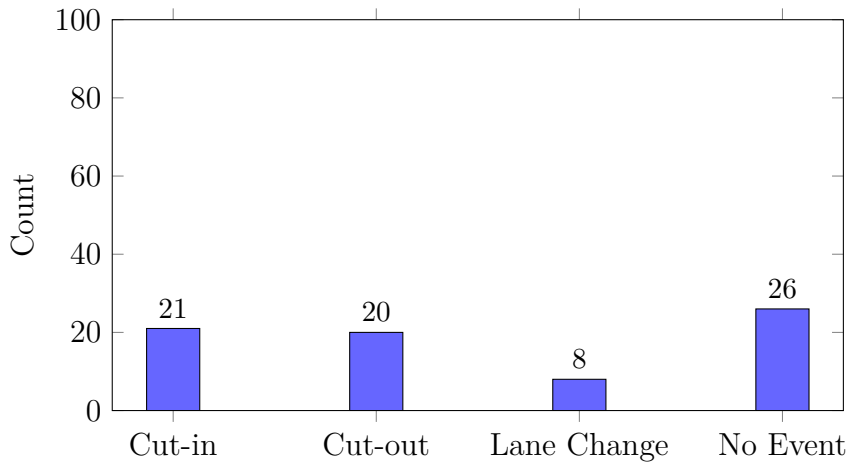


Figure 4.1: Distribution of events in the 75 evaluation sequences.

Using the hyperparameter setup in Table 3.1 with varying values for d_{min} the results are shown in Tables 4.1–4.3.

Scenario	F1 Score	Precision	Recall	FP	FN
Lane Change	0.800	0.857	0.750	1	2
Cut-out	0.706	0.857	0.600	2	8
Cut-in	0.833	1.000	0.714	0	6

Table 4.1: Performance metrics for $d_{\text{min}} = 30$.

Scenario	F1 Score	Precision	Recall	FP	FN
Lane Change	0.800	0.857	0.750	1	2
Cut-out	0.789	0.833	0.750	3	5
Cut-in	0.923	1.000	0.857	0	3

Table 4.2: Performance metrics for $d_{\text{min}} = 50$.

Scenario	F1 Score	Precision	Recall	FP	FN
Lane Change	0.800	0.857	0.750	1	2
Cut-out	0.800	0.720	0.900	7	2
Cut-in	0.878	0.900	0.857	2	3

Table 4.3: Performance metrics for $d_{\text{min}} = 70$.

The interpretation of precision and recall is shaped by how events were annotated. Since the 75 sequences were human-annotated, the annotator's perception of each event directly affects the classifier's measured precision and recall. While cut-ins, cut-outs, and lane changes are conceptually easy to define, they are still difficult to identify and judge from a camera POV. This is relevant because the rule-based classifier applies a distance filter d_{min}, that the human annotators do not. Annotators identify a cut-in or cut-out from the video alone, without knowledge to the distance at which it happens. As a result, it is common for human-annotated events to fall outside the classifier's distance range. These cases reduce recall, but they do not reflect a detection failure. The events simply lie outside the set look-ahead range defined by d_{min}.

Tables 4.1–4.3 show that varying d_{min} affects the performance of our classifier. Increasing d_{min} from 30 to 50 improves recall for both cut-ins and cut-outs. This improvement reflects a closer match between the classifier's range and the distances at which annotators tend to mark events. However, increasing d_{min} further from 50 to 70 shows a trade-off. A larger distance threshold lets in more relevant vehicles at longer ranges. This is relevant because the rule-based classifier operates on a bird's-eye view and has full access to all surrounding vehicles, while the human annotator works from the camera POV, where distant vehicles are often obstructed by other traffic or simply too small to judge reliably. As a result, the classifier detects maneuvers that the annotator never marked, which are counted as false positives despite being true events. Cut-out precision drops from 0.90 to 0.72, with false positives rising from 3 to 7, and cut-in precision shows a similar decline. Thus, d_{min} controls a recallprecision trade-off. Smaller values keep the classifier focused on near-range

events where both views agree, while larger values capture more annotated events at the cost of detections the annotator could not see.

The comparison of the tables show that the lower precision for cut-ins that we achieve in Table 4.2 is a trade-off for a higher likelihood of false positives. If a classifier said "cut-in" but there was no cut-ins in the sequence the model learns to associate that caption with video-frames that don't show the event. False negatives are less damaging. The model learns less, but it doesn't learn something wrong. In context of our purpose, the false positives are more harmful than false negatives.

A known limitation exist for lane-changes. When the lane splits into multiple lanes or combines into one it causes a mismatch between human annotation and our logic. Since the system identifies lane changes by tracking segment transitions, splits pose a challenge. For a split, i.e the road goes from one lane to two lanes, the new segment is encoded as a successor to the previous segment rather than a neighbor. This makes the transitions not recognized as a lane change. Instead it filters it out completely, resulting in a false negative as shown in Figure 4.2.

(a) t_0 (b) t_1 (c) t_2

Figure 4.2: Example of a split causing a false negative.

(a) t_0 (b) t_1 (c) t_2

Figure 4.3: Example of narrow roads causing a false positive.

Another limitation relates to the selection of the current lane segment. The system identifies the closest segment and commits to its centerline until the next recomputation. However, when adjacent lanes are narrow, small inaccuracies in the data or in the distance computation can cause the ego vehicle to oscillate between two

segments, resulting in, and what the logic specifies, a lane change. This is the cause of the single false positive observed, and possible false positives in the larger dataset, illustrated in Figure 4.3.

4.1.2 Application to Full Dataset

The following figures show the distribution of events after applying the rule-based classifier to the full dataset (training and validation splits). These results show how the classifier behaves on the raw data. Recall and precision scores can not be determined on these as they are not manually annotated.

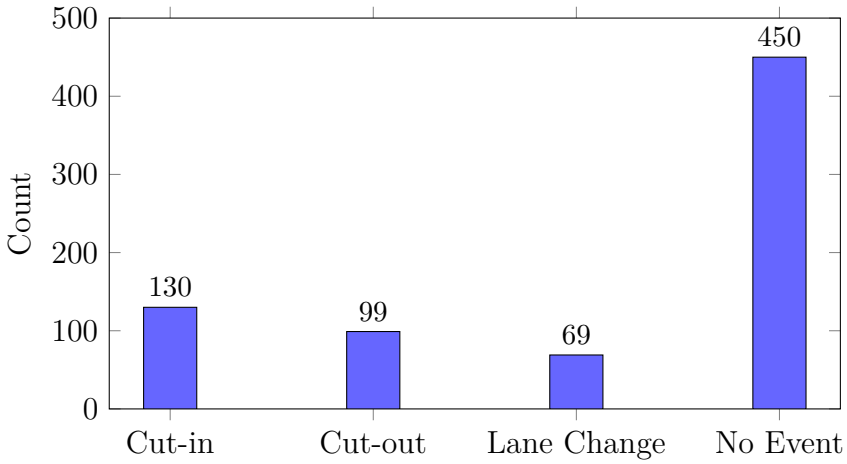


Figure 4.4: Distribution of events in the 700 training sequences.

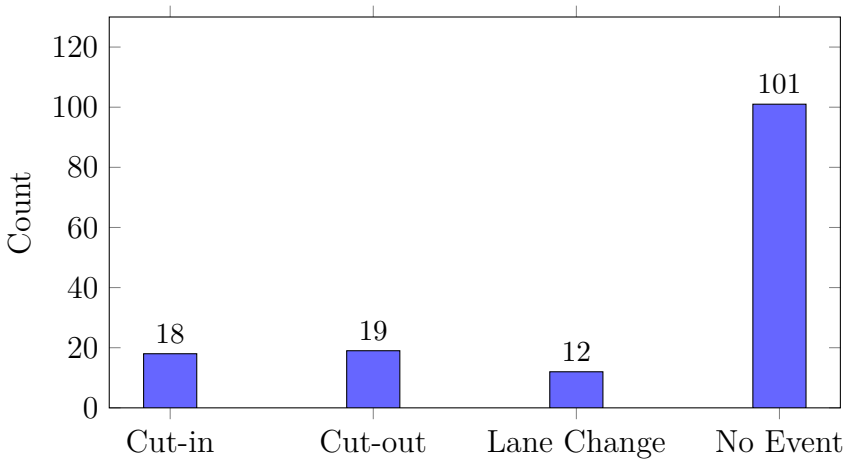


Figure 4.5: Distribution of events across 150 validation sequences.

Figure 4.4 summarizes the distribution of detected events across the 700 training sequences after applying the rule-based classifier. The dataset is imbalanced: 64.3% of sequences contain no detected events, while only 35.7% contain at least one. Among event types, cut-ins are the most frequent, appearing in 18.6% of sequences. Cut-outs appear in 14.1% of sequences, and lane changes in 9.9%. Although the

imbalance is large, it is expected in a real-world driving scenario. The majority of driving sequences are uneventful.

Figure 4.5 summarizes the distribution of detected events across the 150 validation sequences after applying the rule-based classifier. In the validation data 12.0% are cut-ins, 12.7% are cut-outs, 8.00% are lane changes and 67.3% contain no events. These fluctuations are expected given the smaller sample size and do not suggest a meaningful distributional shift between splits.

In comparison to Table 4.1, the human-annotated sequences show a significantly higher number of cut-in and cut-out events than those detected by the rule-based approach. This difference is expected, as the rule-based system applies stricter logic to reduce the risk of false positives.

4.2 Captioning Engine

This section covers the results of the captioning engine and evaluates how design choices affect the generated captions. It also shows examples of captions generated for sequences where a maneuver occurs.

4.2.1 Baseline

This section presents captions generated without the rule-based examples. Using a variant of Figure 3.3 that allows the VLM to describe each scenario without manuever constraints, the captions in Figures 4.6 – 4.8 serve as a baseline for comparison against the rule-based captions.

Cut-in event

Frame 1 Frame 2 Frame 3

Sequence ID: `6aaf5b08-9f84-3a2e-8a32-2e50e5e11a3c`

Caption:

The camera vehicle travels forward on a wet urban street lined with tall buildings under an overcast sky. A dark sedan is directly ahead in the same lane, maintaining a steady distance. To the right, a red bus occupies the adjacent lane, traveling in the same direction. In the left lane, a white SUV is visible ahead, and a black sedan is further back. The traffic flows steadily with no clear lateral movements by any vehicle into or out of the camera vehicle's lane. No vehicle performs a cut-in, cut-out, or lane change that is clearly visible across the frames. The road surface is damp, reflecting the ambient light, and traffic signals are visible ahead, showing green.

Figure 4.6: Cut-in caption generated only by a VLM.

Cut-in and cut-out events

Frame 1 Frame 2 Frame 3

Sequence ID: `2f2321d2-7912-3567-a789-25e46a145bda`

Caption:

The camera vehicle travels on a multi-lane urban road under a clear sky, with tall buildings and palm trees lining the right side. A gray Honda Odyssey is visible in the adjacent left lane, traveling ahead and maintaining its position. The road is marked with white lane lines and features overhead green directional signs for Interstate 95 and 395. A dark pickup truck is seen in the right lane, moving forward and staying within its lane. No vehicle clearly enters or exits the camera vehicle's lane, and no lane markings are crossed by the camera vehicle itself. Traffic flows steadily, with no abrupt maneuvers observed.

Figure 4.7: Cut-in and cut-out caption generated only by VLM.

Lane change event

Frame 1 Frame 2 Frame 3

Sequence ID: `7e4d67b3-c3cc-3288-afe5-043602ea3c70`

Caption:

The camera vehicle travels on a multi-lane urban road under clear, sunny skies. A white police sedan is visible directly ahead in the same lane, maintaining a steady position. To the left, a silver sedan is in the adjacent lane, traveling parallel to the camera vehicle. A black SUV appears in the left lane and moves forward, remaining in that lane without crossing into the camera vehicles path. No vehicle clearly cuts into the camera vehicles lane from an adjacent lane, nor does any vehicle exit the camera vehicles lane to move into an adjacent one. There is no visible lane marking being crossed by the camera vehicle itself, so no lane change occurs. Traffic flows steadily with several vehicles visible ahead, including a yellow taxi and a white van, all maintaining their respective lanes. The scene includes buildings, utility poles, and construction activity in the distance, with no significant changes in traffic behavior or road conditions.

Figure 4.8: Lane change caption generated only using a VLM.

4.2.2 Model Scale

Choosing the model for the VLM was an important design choice in our method. To understand the difference between two different models it is interesting to compare how the choice affect caption quality.

Figure 4.9 compares outputs from the 2B and 30B versions of Qwen3-VL on the same sequence. The 2B model misreads a shopfront as "Fitzgerald's" and repeats it, while the 30B model picks up "Shred-it" on a delivery truck and "Ferlito Construction" on a nearby building, both of which appear in the frames. The 30B caption also grounds itself in the scene more carefully. Instead of a generic list ("a white sedan and a silver SUV ahead"), it points to things that are distinctive to this clip. For example, a mural of a man at a keyboard, the branded truck, the fact that the ego vehicle is following a white SUV.

The large model also sticks closer to the truth. The 2B model describes the road as "dry and well-maintained," which could apply to almost any street. The 30B model notes faded lane markings and cracks in the asphalt. The 2B output also comments itself into a small contradiction, mentioning oncoming traffic on a street where none has been described. Scaling up helped with grounding, but it exposed a different problem. The 30B model still struggled with anything that required temporal reasoning and describing cut-in and cut-out events across frames. When a maneuver happens over time the captions could hallucinate which vehicle does what, and often have tunnel-vision on the closest vehicle in the frame.

Caption Comparison: 2B vs 30B

Frame 1 Frame 2 Frame 3

Sequence ID: `ecbe6def-7560-352c-8822-b2b92613e1e4`

Qwen3-VL-2B

The scene unfolds at a city intersection under an overcast sky, with a grayish-blue hue dominating the horizon, suggesting early morning or late afternoon lighting conditions. The vehicle is approaching the intersection from a straight-ahead direction, with a green traffic light visible at the intersection, allowing for a smooth passage. The road is a multi-lane asphalt street lined with parked vehicles on both sides, including a white sedan and a silver SUV ahead. The surrounding buildings are commercial, with signs for "Fitzgerald's" and "Fitzgerald's" visible, indicating a retail or service area. The road surface appears dry and well-maintained, with clear lane markings. No pedestrians or cyclists are visible in the immediate vicinity, and the traffic is light, with no signs of congestion or active maneuvers. The vehicle is moving steadily forward, maintaining a safe distance from the oncoming traffic, and no significant maneuvers are detected in this sequence.

Qwen3-VL-30B-A3B

The camera vehicle travels through an urban street under an overcast sky, with a multi-lane road marked by faded white lane lines and visible cracks in the pavement. On both sides of the street, vehicles are parked along the curbs, including sedans, SUVs, and a white delivery truck with "Shred-it" branding. A large mural of a man playing a keyboard is painted on the side of a building on the left, while a "Ferlito Construction" sign is visible on a building to the right. Traffic flows steadily ahead, with a white SUV directly in front, and the camera vehicle follows behind. The traffic lights are green, allowing continuous movement through the intersection and along the street. No significant maneuvers such as cut-ins, cut-outs, or lane changes are detected during the sequence. The scene is illuminated by streetlights and the dim light of the cloudy day, with no pedestrians or cyclists visible.

Figure 4.9: Comparison of captions generated by two model scales for the same sequence.

4.2.3 Adding Context to the Prompt

Adding additional context in the prompt for the VLM affects the quality of the caption. Additional context are things like telling VLM explicit details, for example what vehicle made the cut-in and when it happen in the sequence, if it was in the beginning, middle or the end. Figure 4.10 displays how the additional information improved temporal reasoning.

The results shows that prompt engineering has strong impact on the quality of the output. Without additional context, the model often identifies that maneuvers occur but often fails to track with vehicle performs them. With added context, the captions became more accurate and consistent. The improvement is most visible in sequence where multiple maneuvers happen. Another common failure of the default context was hallucinating that the vehicle closest to ego always performed the detected maneuver. With the added context the VLM correctly identifies which vehicle is doing what. It also helps the VLM distinguish between multiple vehicles in a sequence rather than focusing and getting tunnel-vision on just one.

Caption Comparison: Adding context to the caption

Frame 1 Frame 2 Frame 3

Sequence ID: `0b86f508-5df9-4a46-bc59-5b9536dbde9f`

Qwen3-30B - default

The camera vehicle travels along a multi-lane urban road under a clear blue sky, with palm trees lining the right side and high-rise buildings visible in the distance. A silver sedan is in the adjacent left lane, while a dark SUV is directly ahead in the same lane. As the vehicle proceeds, the silver sedan cuts into the camera vehicle's lane from the left, completing a cut-in maneuver. Later, the dark SUV ahead performs a cut-out, moving into the right lane and exiting the camera vehicle's path. The road features a dedicated bike lane marked with painted symbols, and construction activity with orange cones is visible on the right shoulder. Traffic flows steadily, with other vehicles, including trucks and sedans, present in adjacent lanes and further ahead. The scene is illuminated by bright daylight, with overhead signs indicating directions for Interstate 195 East and State Road 27 North.

Qwen3-30B - enriched

On a sunny day with clear blue skies, the camera vehicle travels in a multi-lane urban road lined with palm trees and modern buildings. A green overhead sign indicates directions for Interstate 195 East and Route 27 North. Traffic flows steadily, with a silver sedan visible in the adjacent left lane and a dark SUV ahead in the same lane. A short distance ahead, a sedan or SUV (Vehicle A) performs a cut-in, moving from the adjacent lane into the camera vehicle's lane. Shortly afterward, the same vehicle (Vehicle A) executes a cut-out, leaving the camera vehicle's lane and merging into the adjacent lane. The road features a painted bicycle lane and a speed limit sign indicating 30 mph. In the distance, high-rise buildings and construction activity are visible along the corridor.

Figure 4.10: Comparison of captions when being enriched with additional details.

4.2.4 Caption Examples

This section shows a example of a caption the models produces on the three events we covered: cut-in, cut-out, and lane change. The model describes both the manuever that happend and the environment. Figure 4.12- 4.13 display examples of captions produced for the three different event-types.

Cut-in caption

Frame 1 Frame 2 Frame 3

Sequence ID: `6aaf5b08-9f84-3a2e-8a32-2e50e5e11a3c`

Caption:

The camera vehicle travels on a wet urban street under overcast skies, with tall buildings lining both sides of the road. A dark sedan is directly ahead in the same lane, and a red bus is visible in the adjacent right lane. As the camera vehicle proceeds, the red bus moves from the right lane into the camera vehicle's lane, performing a cut-in maneuver. The bus continues forward, now occupying the same lane as the camera vehicle. Traffic flows steadily, with other vehicles visible in the distance and a pedestrian crossing the street ahead. The road surface is reflective from recent rain, and traffic signals are visible at the intersection ahead.

Figure 4.11: Example of a cut in caption.

Cut-in and cut-out caption

Frame 1 Frame 2 Frame 3

Sequence ID: `2f2321d2-7912-3567-a789-25e46a145bda`

Caption:

The camera vehicle travels on a multi-lane urban road under a clear sky, with tall buildings and palm trees lining the right side. A gray Honda Odyssey is visible ahead in the same lane, positioned directly in front of the camera vehicle. As traffic flows forward, the Odyssey begins to move from the adjacent left lane into the camera vehicle's lane, completing a cut-in maneuver. Shortly after, the same vehicle initiates a cut-out, shifting right into the adjacent lane and leaving the camera vehicle's path. The maneuver occurs under an overpass with overhead green highway signs indicating directions to Interstate 95 and 395 West. Other vehicles, including sedans and SUVs, are present in adjacent lanes, maintaining steady traffic flow. The road surface is dry, and lane markings are clearly visible.

Figure 4.12: Example of a cut-in and cut-out caption.

Lane change caption

Frame 1 Frame 2 Frame 3

Sequence ID: `7e4d67b3-c3cc-3288-afe5-043602ea3c70`

Caption:

The camera vehicle is traveling on a multi-lane urban street under a clear blue sky, with a white police car visible ahead in the right lane. A silver sedan is directly in front, and a black SUV is in the adjacent left lane. The camera vehicle executes a lane change to the left, moving past the silver sedan and positioning itself alongside the black SUV. Traffic flows steadily, with a yellow taxi and a white van visible further ahead. On the right, a tall white building with balconies lines the street, and outdoor seating is set up at a sidewalk caf̆00e9. Utility poles and overhead wires run along the road, and construction activity with a crane is visible in the distance. The road surface is paved and marked with white lane lines, and the scene is illuminated by bright daylight.

Figure 4.13: Example of a lane change caption.

4.3 Fine-Tuning CLIP and PE

In the following subsections we compare visual backbones and their baselines, as well as the effect of caption richness on validation accuracy. Each evaluated on a held-out validation set of 150 dash-cam sequences.

4.3.1 Baselines

A baseline for both backbones were computed and evaluated on the same validation set with metrics displayed in Table 4.4. PE-Core-L14-336 outperforms CLIP-ViT-L/14-336 on almost every retrieval metric out of the box.

	CLIP ViT-L/14-336		**PE-Core-L14-336**	
Metric	T→V	V→T	T→V	V→T
R@1	25.0%	28.9%	26.3%	30.3%
R@5	52.0%	60.5%	55.3%	57.2%
R@10	64.5%	73.7%	70.4%	69.1%
Median Rank	4	2	3	3
Alignment	1.4884	—	1.4845	—
Uniformity (vid)	−0.6356	—	−0.9074	—
Uniformity (txt)	−0.8774	—	−0.9858	—

Table 4.4: Zero-shot baseline metrics.

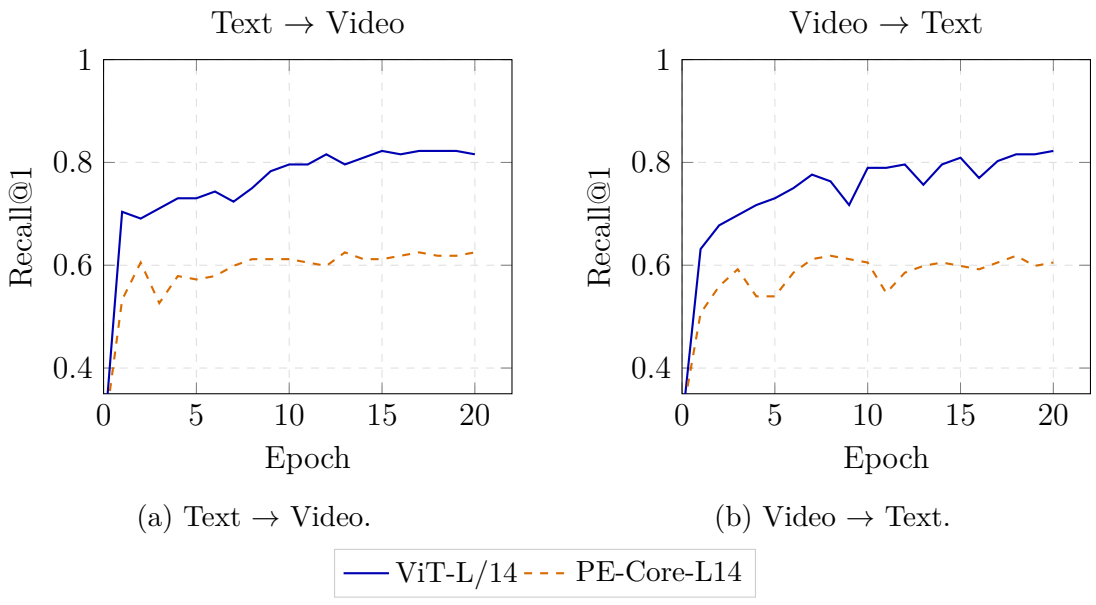
4.3.2 Comparison of CLIP and PE

We fine-tune both backbones under identical training settings (see Table 4.5) and compare retrieval performance on the held-out validation set.

Table 4.5: Training hyperparameters used for both backbones.

Hyperparameter	**Value**
Image size	336×336
Pretrained weights (only CLIP)	OpenAI
Optimizer	AdamW
Learning rate	5×10^{-6}
Weight decay	0.01
Warmup fraction	0.05
Gradient clip (max norm)	1.0
Epochs	20
Batch size	8
Frames per video	4
Frozen visual transformer blocks	8
Frozen text transformer blocks	8
Rotation (± deg.)	5
Brightness jitter	0.3
Contrast jitter	0.3
Saturation jitter	0.0

Figure 4.14 shows that CLIP (ViT-L/14) outperforms PE-Core (L14-336) across all training epochs in both retrieval directions.

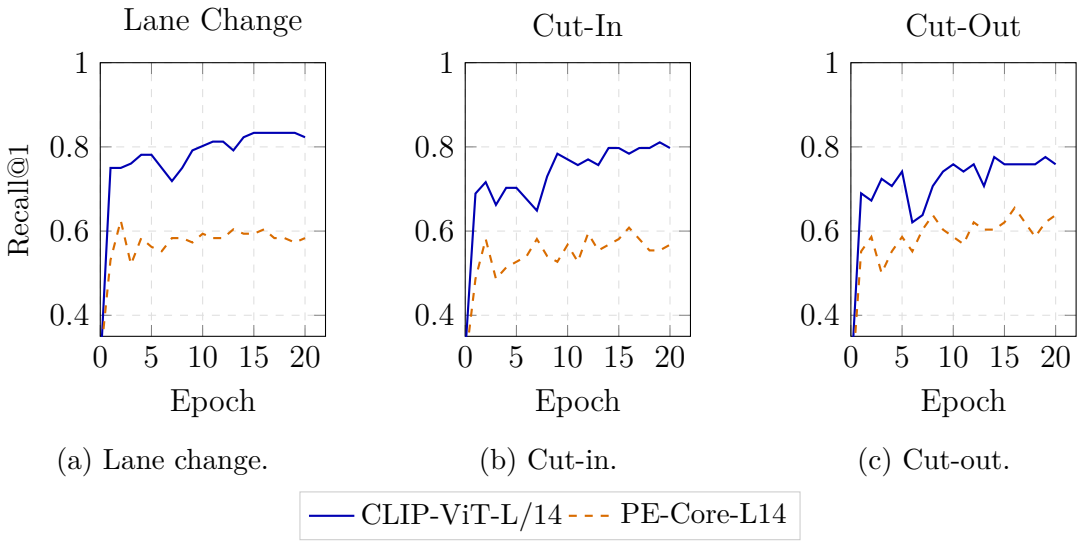


(a) Text → Video.

(b) Video → Text.

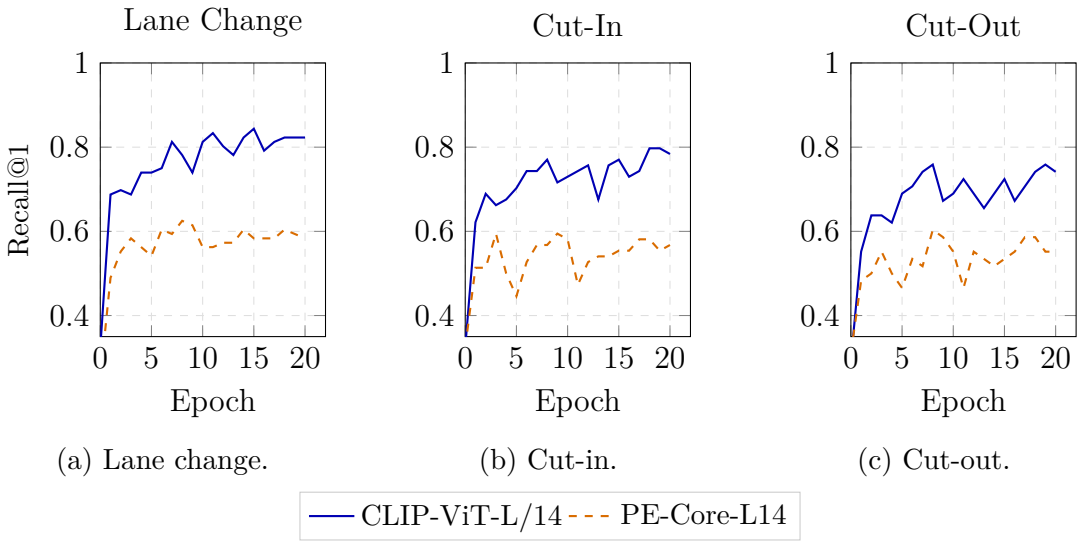
Figure 4.14: Overall Recall@1 over training epochs for both backbones.

The per-event breakdown in Figures 4.15 and 4.16 shows that CLIP is stronger in each respective maneuver type. Lane-change retrieval is the strongest category for both backbones, while cut-out events are the hardest for both.



(a) Lane change. (b) Cut-in. (c) Cut-out.

Figure 4.15: Text → Video Recall@1 across manuever types.



(a) Lane change. (b) Cut-in. (c) Cut-out.

Figure 4.16: Video → Text Recall@1 across manuever types.

4.3.3 Effect of Prompt Engineering

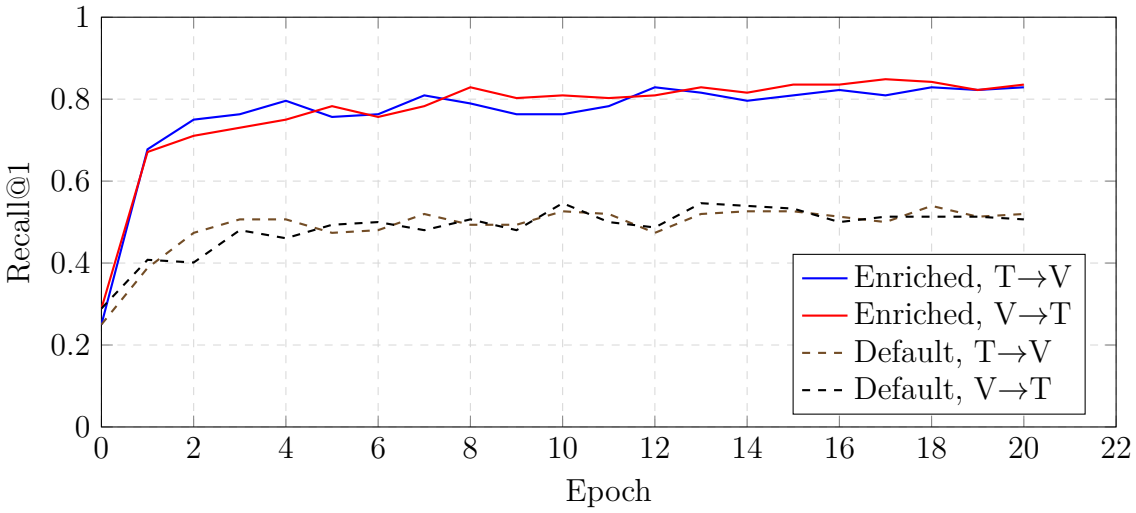
The captioning pipeline produces two variants of the training data: *default* captions, generated by Qwen3-VL-30B from the fact that there are events in the sequence given by the rulebased classifier, and *enriched* captions, where the same model is given additional context, such as timestamps of events and vehicle ids. To isolate the effect of this enrichment, we fine-tuned the same backbone with the same hyperparameters on each caption set in turn.

Final-epoch retrieval metrics for both runs are summarised in Table 4.6. Caption enrichment improves Text→Video Recall@1 from 52.0% to 82.9% and Video→Text Recall@1 from 50.7% to 83.6%. This effect is shown across all R@K levels and across all three maneuver types.

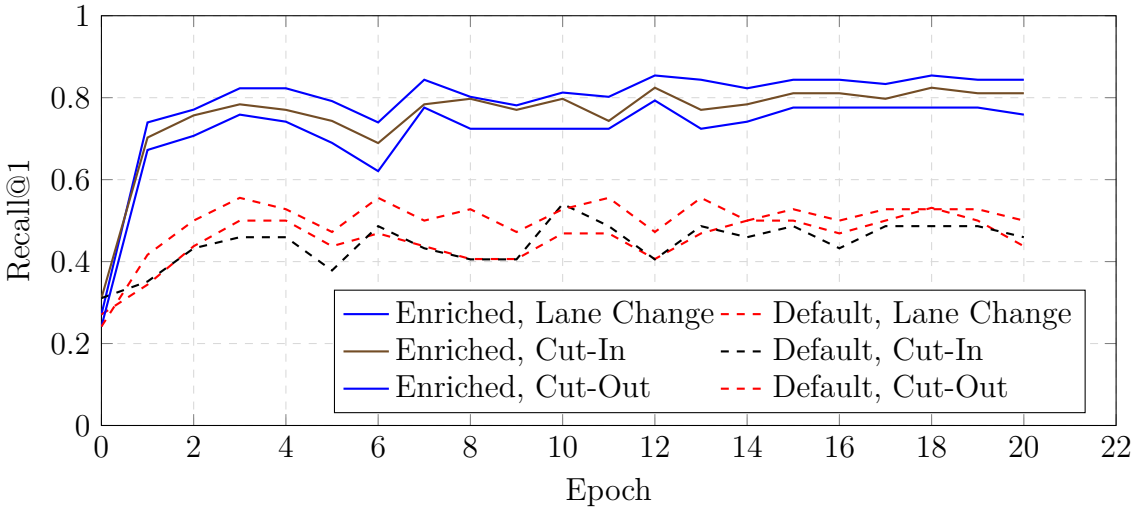
Metric	**Default**	**Enriched**	**Δ**
R@1 (T→V)	52.0%	82.9%	+30.9 pp
R@5 (T→V)	82.2%	96.7%	+14.5 pp
R@10 (T→V)	89.5%	99.3%	+9.8 pp
R@1 (V→T)	50.7%	83.6%	+32.9 pp
R@1 Lane Change	43.8%	84.4%	+40.6 pp
R@1 Cut-In	46.0%	81.1%	+35.1 pp
R@1 Cut-Out	50.0%	75.9%	+25.9 pp

Table 4.6: Final-epoch retrieval metrics for fine-tuning with and without caption enrichment.

The largest per-maneuver gain from enrichment is for Lane Change, followed by Cut-In and Cut-Out. The event-type ordering also changes between the two runs. Without enrichment, Cut-Out is the easiest maneuver to retrieve, while with enrichment Lane Change is the easiest.



(a) Overall Recall@1, both runs, both retrieval directions.



(b) Text→Video Recall@1 across maneuver types.

Figure 4.17: Effect of caption enrichment on retrieval performance during fine-tuning of CLIP.

5

Discussion

The rule-based classifier was tuned to reach a precision-recall trade-off suitable for the project scope. Setting $d_{\min}$ to 30 minimised false positives, which was the priority for our pipeline. As consequence of the class distribution shown in Figure 4.4- 4.5 we have to be mindful of how we value false positives versus false negatives. This class imbalance amplifies the cost of label noise, because errors are concentrated on the underrepresented classes where the model has the least clean signal to begin with. As Graham-Knight et al. state, when minority classes contain mislabeled samples, models may treat hard-to-learn but clean class samples as noise and disregard genuine examples from those classes [21]. In our case, this means a false-positive maneuver detection on a rare class such as lane changes, cut-outs and cut-ins are more harmful. Since it creates an incorrect video- caption pair, and that pair's relative weight within the small minority partition is large. More false positives make the downstream model more likely to learn the wrong associations. With more false negatives we simply get less training data, but also less noisy training. However, there is a trade-off, since the class imbalance is large we cannot have many false negatives either. Since, the more false negatives the less training pairs we acutally get.

One limitation of our evaluation on the rulebased classifier is the way the human annotations were produced. The annotators worked only from image data, without access to map- or sensor information. Further, they only used observations from the front-facing camera. This creates two issues. First, events flagged by the rule-based classifier from sensor data may not be visible to the annotator, for instance when another vehicle is obstructed by traffic or falls outside the camera's field of view. The classifier and the annotator are, in such a case, looking at different inputs. Second, the lack of side-camera footage means events occurring next to the ego vehicle is completely missed by the annotator but can still be detected from sensor data. Visual degradation is also a concern, for example blur or partial obstruction. These are often easy for a human to flag, but tends to be overlooked by VLMs if not explicitly prompted for it, as previous research has shown [9].

This labelling and annotation discrepancy could partly explain the distributional gap between Figures 4.1 and 4.4. The annotated set shows a higher event rate than the rule-based output on the larger dataset. Additioanlly, the precision-recall trade-off contributes as well, since a stricter $d_{\min}$ filters out events the human would still mark.

A better evaluation setup would either use multiple camera angles for annotation or that the annotator had a map view with explicit distance information, removing d_{min} from having such a large impact. Moreover, a larger annotated set would also give a clearer picture of the classifier's behavior. But manual labeling was too time-consuming to annotate more samples within the project's time-frame.

When it comes to baselines, PE-Core outperforms CLIP on almost all retrieval tasks (Table 4.4). This is similar to what Bolya et al. [7] show, that PE outperforms CLIP on a broad range of zero-shot image and video classification and retrieval benchmarks. A more surprising outcome is that after fine-tuning, CLIP outperforms PE-Core across all retrieval metrics and all three maneuver types. One possible explanation is the limited size of our domain-specific dataset. While the sliding window expands 700 dash-cam sequences into approximately 4200 sub-sequences, these are largely repeated views of the same underlying scenes and do not provide the visual diversity that a larger model like PE-Core typically benefits from during fine-tuning. The properties that make PE strong as a zero-shot model do not necessarily translate into an advantage when fine-tuning on this kind of small, low-diversity subset. Thus, a more diverse and larger dataset would likely benefit both model back-bones.

A second possibility is that the two encoders were not equally suited to the chosen training configuration. Both backbones were fine-tuned under identical hyperparameters (Table 4.5) to ensure a good comparison, but identical hyperparameters do not guarantee correct optimality. However, the different hyperparameters tried on CLIP were also tried on PE-Core, with CLIP perfoming better in all experiments. One thing to note is that the recall change over epochs in Figure 4.14 show that PE-Core's curve is flatter. This could mean that the model is not as prone to move away from its strong pretrained initialization. A learning-rate sweep was outside the scope of this thesis but is a natural next experiment.

The captioning enrichment also had a big impact on our downstream results. Combining structured rule-based description with a VLM have shown in previous research to improve captions that are grounded in the driving scenarios [2], [18]. This is also in line with what Lai et al. [22] show. Rewriting image captions to include more detailed descriptions give substantial improvement in CLIP retrieval performance. Although, the auto-motive setting and the use of fine-tuning instead of pre-training is different in our purpose compared to Lai et al's the underlying mechanisms appear to be the same. Captions that are better grounded in the visual content of the paired image provide a better signal for the contrastive objective. This is relevant given the well-documentated limitation that VLMs alone lack the spatial grounding needed for capture detailed attributes of driving scenarios [1], [2], [3], [18].

This limitation is visible in our own baseline captions. Without rule-based grounding the VLM frequently fails to register that any maneuver has occurred [9]. In Figure 4.6, the caption describes a red bus in the adjacent lane and describes that there were "no clear lateral movements," when the bus does in fact cut into the ego lane. A similar pattern can also be seen in Figure 4.7–4.8, where the model tend to

default to "traffic flows steadily" and "no abrupt maneuvers observed.".

Adding the rule-based classifier makes the captioner more focused. The rule-based system supplies the event label, the involved vehicle, and when the event occurs, leaving the VLM to ground this information in the dash-cam sequence. The result is that captions more consistently capture the maneuver in each sequence, instead of defaulting to similar descriptions of traffic flow. This better temporal grounding with the good environment descriptions gives the contrastive objective a clear signal to align on. This is likely the driver of the better retrieval performance seen in Table 4.6.

While caption enrichment improved retrieval performance, several aspects of the prompting strategy were not fully explored. Now, the captions produced by Qwen3-VL-30B-A3B-Instruct were typically four to eight sentences long, which in many cases contains more detail than is needed to describe the maneuver. Our template (Figure 3.3) forces the model to include description of environment and surrounding vehicles in every caption, even if its not clear that there is valuable information in the sequence. In some sequences, this caused the model to hallucinate details that were not present or produce captions that were far too lengthy for what the sequence actually included. This is a known failure of VLMs to fabricate captions when prompts are asking for more information than what the visual evidence actually supports [9]. Thus, future experiments could focus on producing more concise captions and compare them directly to the enriched ones. It is plausible that more concise captions could reduce hallucinations and provide a cleaner training signal for the downstream VLM model.

Another limitation of our captioning pipeline is that we have no robust way to evaluate the quality of the captions themselves. Throughout our work, quality has been assessed by manual examination on a subset of sequences. We observe that enriched captions improve downstream retrieval, but this is an indirect signal. A caption can boost retrieval simply by mentioning one of the keywords related to cut-ins, cut-outs, or lane changes, even if the surrounding scene description is hallucinated. Similarly, a more grounded and truthful caption could underperform on retrieval if it contains fewer such keywords. Diagnosing where the VLM succeeds or fails is therefore a difficult task, especially since we have no way to evaluate all sequences manually. There are, however, strategies that could address this. One approach is to a large multi-modal language model (MLLM), such as Claude or ChatGPT, as a pseudo-ground-truth annotator. Although, MLLM's also struggle with complex temporal dynamics in videos [23], the structured maneuver information from our rule-based classifier could be injected into the prompt to anchor the scene description. The captions produced by the VLM could then be compared to these larger models' captions using NLP similarity metrics such as BLEU or BERTScore to compute token overlap or semantic similarity [24], [25]. That said, such a setup would introduce its own difficulties and warrants its own study.

All of the improvements and implementations of above essentially boils down to a downstream model where R@1 jumps from 52% to 83% (in the text-to-video direction). This naturally raises the question: is the fine-tuned model performing

fine-grained scene retrieval, or is it learning a four-way classification over maneuver categories? Both default and enriched captions name the maneuver type, which is the strongest discriminative signal in the text. The fact that the largest maneuver-type gain is for lane change, which is the rarest class, is consistent with the model learning to attend more to maneuver keywords when they are placed in a more correct context. For example, the default captions included the maneuver but the enriched described everything else surrounding the maneuver (as well as the maneuver itself) more accurately. For the use case of curating edge cases this behavior is what we want, but it makes the result harder to read as evidence of better visual-textual alignment. A useful follow-up would be to mask the maneuver word from the caption at evaluation time and see how much of the gain holds up.

Finally, both training and validation sets are heavily skewed toward no-event sequences (74.7% in validation), which inflates aggregate retrieval metrics relative to the practical goal of locating rare maneuvers. The per-event breakdowns in Figures 4.15 and 4.16 are therefore the more meaningful evaluation for a curation use case, and the overall Recall@1 numbers should be read as an upper bound for querying for specific maneuver type.

6

Conclusion

This thesis aimed to develop a captioning engine and a representation-learning framework for autonomous driving data, with the goal of supporting text-based retrieval of driving scenarios for dataset curation. Two objectives were addressed: building a captioning engine that uses rule-based classifier to produce rich and grounded captions, and fine-tuning a contrastive vision-language embedding model on the resulting video-caption pairs to improve retrieval performance.

The results show that Recall@1 improved from 25% in the zero-shot baseline to 83% after fine-tuning on enriched captions. Most of this increase comes from the metadata enrichment itself. The default rule-based system reached only 52% Recall@1 after fine-tuning, but when enriching the captions with metadata about the maneuver the downstream retrieval tasks increased by approximately 30 percentage points on its own. The remaining findings can be summarized along three axes. Firstly, the rulebased classifier achieved F1 scores between 0.80 and 0.92 across the maneuver types at $d_{\text{min}} = 30$ on a manually annotated set of sequences. Second, caption quality depends on the chosen model as our results show that the 30B parameter model of Qwen3-VL produced more consistent and grounded captions than the 2B model did. Third, on the contrastive backbones, CLIP outperformed PE-Core consistently across all retrieval metrics after fine-tuning for all maneuvers that were analyzed.

Several directions remain open for future research. The rule-based classifier could be extended to include more maneuver types such as lead-vehicle braking and vulnerable-road-user interactions for understanding if representations can also be learned for those cases. Different and larger datasets could be used to evaluate whether the learned representations transfer to other domains. On the captioning side, shorter and more focused captions should be compared against the enriched captions to understand if lengths adds value or it just increases the model's tendency to hallucinate details. Finally, the fine-tuning of the encoders should be investigated further to determine if the performance of PE-Core can be improved.

Taken together, the work shows that combining a rule-based classifier with captions produced from a VLM produces enough signal to fine-tune a contrastive vision-language embedding model for driving scenarios. Text-based retrieval of driving scenarios is therefore a viable foundation for scalable dataset curation.

Bibliography

[1] C. Cui et al., "A Survey on Multimodal Large Language Models for Autonomous Driving," *arXiv:2311.12320*, 2023.

[2] V. Gopinathan, U. Zimmermann, M. Arnold, and M. Rottmann, *Temporal object captioning for street scene videos from lidar tracks*, May 2025. DOI: 10.48550/arXiv.2505.16594.

[3] X. Tian et al., "DriveVLM: The Convergence of Autonomous Driving and Large Vision-Language Models," *arXiv:2402.12289*, 2024.

[4] Zenseact, *Zenseact research*, Accessed: 2026-05-06, 2026. [Online]. Available: https://research.zenseact.com/.

[5] X. Zhou et al., "Vision Language Models in Autonomous Driving: A Survey and Outlook," *IEEE Transactions on Intelligent Vehicles*, 2024.

[6] A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[7] D. Bolya et al., "Perception Encoder: The Best Visual Embeddings Are Not at the Output of the Network," *arXiv:2504.13181*, 2025.

[8] M. M. Antony and R. Whenish, "Advanced Driver Assistance Systems (ADAS)," in *Automotive Embedded Systems*, ser. EAI/Springer Innovations in Communication and Computing, M. Kathiresh and R. Neelaveni, Eds., Cham: Springer, 2021. DOI: 10.1007/978-3-030-59897-6_9.

[9] S. Xie et al., "Are VLMs Ready for Autonomous Driving? An Empirical Study from the Reliability, Data, and Metric Perspectives," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[10] S. J. D. Prince, *Understanding Deep Learning*. MIT Press, 2023.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[13] M. Gheini, X. Ren, and J. May, "Cross-Attention is All You Need: Adapting Pretrained Transformers for Machine Translation," *arXiv:2104.08771*, 2021.

[14] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General Perception with Iterative Attention," *arXiv:2103.03206*, 2021.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," *arXiv:2002.05709*, 2020.

[16] J. Yu, J. Wang, Z. Vasudevan, et al., "CoCa: Contrastive Captioners are Image-Text Foundation Models," *arXiv:2205.01917*, 2022.

[17] F. Cui, Y. Zhang, X. Wang, X. Wang, and L. Xiao, "Generalizable Prompt Learning of CLIP: A Brief Overview," *arXiv:2503.01263*, 2025.

[18] Q. Zhao et al., "CoVLA: Comprehensive Vision-Language-Action Dataset for Autonomous Driving," *arXiv:2408.10845*, 2024.

[19] B. Wilson et al., "Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting," in *Proceedings of the NeurIPS Track on Datasets and Benchmarks*, 2021.

[20] A. Yang et al., "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[21] J. B. Graham-Knight, J. Fayyad, N. Bayasi, P. Lasserre, and H. Najjaran, "Conformal-in-the-Loop for Learning with Imbalanced Noisy Data," *arXiv:2411.02281*, 2024.

[22] Z. Lai et al., "VeCLIP: Improving CLIP Training via Visual-enriched Captions," *arXiv:2310.07699*, 2023.

[23] A. Rasekh, E. B. Soula, O. Daliran, S. Gottschalk, and M. Fayyaz, "Enhancing temporal understanding in video-llms through stacked temporal attention in vision encoders," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. [Online]. Available: `https://arxiv.org/abs/2510.26027`.

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002, pp. 311–318. [Online]. Available: `https://aclanthology.org/P02-1040/`.

[25] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: `https://arxiv.org/abs/1904.09675`.