Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Stationary Vehicle Classification Based on Scene Understanding

Weitian Wang

Course of Study:	Computer Science
Examiner:	Prof. Dr. Andreas Bulling
Supervisor:	Dr. Thomas Michalke Dr. Lei Shi
Commenced:	August 22, 2023
Completed:	February 22, 2024

Abstract

Navigating through dense traffic situations like merging onto highways and making unprotected left turns remains a challenge for the existing autonomous driving system. Classifying vehicles into parked, stopped, and moving vehicles can benefit the decision-making system in this case because they play different roles during the vehicle-to-vehicle negotiation process. Existing works in vehicle classification focused on trivial cases and used methods that are not generalized enough. To fill this gap, after analyzing this problem and summarizing the necessary information needed for this problem, we propose a multi-modal model that can leverage information from lidar, radar, camera, and high-definition maps. To meet the complexity of our task and the needs of our model, we collect the dataset in real driving scenario and then preprocess and label it. By utilizing a pretrained vision encoder for fine-grained visual feature extraction and vision foundation model (CLIP) for scene understanding, our model achieves a 97.63% test accuracy on our dataset. Through visualization methods, experiments, and quantitative analyses, we investigate the effectiveness and importance of different encoders used in our model. We interpret and explain the successes and failures of our model to give a better understanding of how different latent features contribute to the final result. In the end, the limitations of our model and potential improvements are discussed.

Contents

1 Introduction

In spite of recent advancements in autonomous driving solutions[HYC+23; NAZ+22], navigating through dense traffic situations like merging onto highways and making unprotected left turns remains a challenge for the widespread deployment of autonomous vehicles[SAR18]. Driving in dense traffic conditions is intrinsically an interactive task[UGA+15], where the autonomous vehicles' actions elicit immediate reactions from nearby traffic participants and vice-versa. Making safe, intelligent, and socially compatible decisions is one of the core capabilities for autonomous vehicles targeting widespread deployment in the real world[HWL21].

To meet this challenge, there have been many works in the field of interaction-aware prediction[HLW+23; KVB22]. However, most of these works are based on labeled datasets (e.g. inD[BKM+20]) where parked vehicles are already distinguished and excluded from the decision-making process. In the real world, autonomous driving systems need to do this kind of classification on their own. Depending on the scenario, a wrong belief in the parked vehicle might result in a dangerous overtaking maneuver and furthermore in blockage of the traffic flow or even in a collision risk.

Further than the binary classification of vehicles into parked and non-parked vehicles, identification of the stopped vehicle can also be beneficial. On one hand, stopped vehicles should be distinguished from parked vehicles because of potential dangers coming from unpredictable driver's behaviour. On the other hand, stopped vehicles should be distinguished from moving vehicles because stopped vehicles play a different role than moving vehicles during the vehicle-to-vehicle negotiation. At the intersection, stopped vehicles that are waiting are expected to be driven by while a moving vehicle in the process of a left turn may expect the ego vehicle to yield. For stopped vehicles pulled over to pick up or set down persons, overtaking behaviour is expected.

In this work, we focus on the task of distinguishing between parked, stopped, and moving vehicles. A single modality is insufficient to tackle this problem. For example, it is hard to distinguish a stopped vehicle waiting at the intersection from a parked vehicle with radar. Ideally, it is possible to only rely on the camera, as human drivers rely on their eyes. However, this is only possible because human eyes are very powerful vision and depth detectors and human drivers have strong reasoning ability that comes from their experience.

To consider what kind of information is relevant for this task and should be included in our model, we begin by examining how humans typically make this kind of decision. When a human driver sees a nearby vehicle and tries to evaluate its motion status, they might ponder the following questions implicitly:

- Is there a driver inside?
- Are the brake lights on?
- What kind of car is it?

- Are there people next to the car?
- Is this a parking area?
- Is here a controlled-access road?
- What is the former states (velocity, position) of the car?
- Are there obstacles / pedestrians in front of the car?

Human drivers do this kind of classification of surrounding vehicles implicitly while driving based on their experience and understanding of the surrounding environment. From the perspective of an autonomous driving vehicle, these questions can be answered by multi-modal information from different sensors.

Inspired by this, we propose a multi-modal classifier that can leverage information from lidar, radar as well as camera and high-definition maps that can achieve 97.63% accuracy on our dataset. Our model consists of four modality encoders: states encoder, global vision encoder, object-level vision encoder, and map encoder. The idea is to better leverage information from different sensors and combine them for vehicle classification. Our model first takes into consideration the tracked history of the target vehicle along with nearby traffic participants. Then, the model evaluates the object and the surrounding scene visually through multi-scale image crops that are encoded with pretrained foundation models. Map information is encoded with a convolutional neural network to provide additional information about the target with respect to the road from a bird's eye view. The latent features given by four encoders are fused through concatenation and decoded by a linear layer. We also conducted experiments on the impact of each modality and visualized their attention or weight distributions.

Existing large-scale datasets[GLSU13; WQA+23] for autonomous driving don't have annotations for stopped vehicles. To fill this gap, we collect the first dataset that includes labels for the vehicles' motion status (parked, stopped, moving) in dense traffic scenarios that includes various cases of stopped vehicles. Our dataset consists of long-time data sequences taken with vehicles equipped with various sensors including high-resolution cameras, lidar, radar, accelerometer, and GPS.

Our contributions can be summarized as follows:

- We evaluate the limitations of existing work in the field of vehicle motion state classification by introducing the "stopped" label and including more complex or ambiguous cases in the dataset.
- We present a transformer-based multi-modal model that can fuse the information from various sensors about the target vehicle and the surrounding scene into a latent fused feature that can be used for vehicle classification.
- We incorporated vision foundation models in our model for scene understanding and have conducted thorough experiments on the choice, the attention distribution, and the effect of vision foundation models under the context of vehicle classification.

2 Related Work

2.1 Vehicle Classification for Autonomous Driving

Despite being a very important and meaningful component of traffic prediction, little work has addressed the vehicle motion state classification problems. Behrendt et.al.[BMBL19] first introduce the problem of parked car classification in the context of autonomous driving. They introduce a list of possible sensors and features, e.g., estimated velocity, distance to left/right lane boundaries. Out of the 13 candidate features they selected, they use the best 5 features and present three baseline approaches based on heuristic thresholding, support vector machines (SVMs), and a simple multilayer perceptron (MLP).

Their main limitations lie in several aspects. First, they used manually selected features that heavily rely on feature engineering which makes their work hard to be generalized well on different datasets. For example, their most important feature is "distance to the left lane boundary". However, the road width of different areas varies a lot. It is impossible for the model to learn a criterion that suits them all.

Second, they didn't include visual information, which is the most important information when humans analyze the motion state of a vehicle. Without visual information about the vehicle and the surrounding scene, non-trivial cases discussed here[BMBL19], e.g., bus loading passengers, can never be properly evaluated by the model.

What's more, They merely focused on the binary classification of parked and moving(with driver inside) vehicles without further distinguishing stopped vehicles from moving vehicles or parked vehicles. Also, they use a small and unbalanced dataset with trivial cases.

Yang et.al.[YLLS21] used a similar architecture where 10 selected features are fed into a multilayer feedforward network to generate a binary output. Differently, they directly focused on the overtaking decision and used whether the target vehicle is overtaken by the ego vehicle as the predicted label. They take the human driver's decision as the target label in their dataset which is recorded during data collection to teach the model to make more human-like decisions. For example, the second most influential feature is the waiting time, because human drivers tend to make overtaking decisions following a period of waiting.

2.2 Motion forecasting

Motion forecasting is the task of predicting the location of road agents (vehicles, pedestrians, etc.) in the future[NAZ+22]. Motion forecasting mainly focuses on constantly moving targets instead of stopped targets because stopped targets are unpredictable. Hence, works in motion forecasting

can't be used as a direct solution to vehicle motion state classification. However, since the sensors available on the vehicle are the same for both tasks, their approaches for multi-modal scene encoding are still worthy of reference.

The state of art in motion forecasting encodes features including agent history, agent interactions, map information, and traffic light states[NAZ+22]. These features are usually encoded and fused into a latent feature for downstream tasks. There are many different mechanisms of doing that, in this work, we will use a simple multi-modal encoder that is discussed in detail in Section 3.1.

2.3 Transformer-based models in autonomous driving

This section follows Zhong et. al.'s excellent survey [ZLC23].

Attention mechanism [BCB16] has been introduced to further enhance the performance of CNN-based methods in autonomous driving. The primary idea behind attention mechanisms is to allow the model to weigh different parts of the input based on their relevance to the current context. In the case of sequence-to-sequence models, for example, attention mechanisms enable the decoder to focus on specific parts of the source sequence while generating the target sequence. This selective focus allows the model to capture longer-range dependencies and relationships more effectively than traditional recurrent neural networks. The attention mechanism can be described as a function that computes a weighted sum of a set of input values, also known as "values" (V), based on their compatibility with a given query (Q). The attention mechanism calculates an attention score for each key-query pair using an attention scoring function and then normalizes these scores using a softmax function to produce attention weights. These weights are used to compute the weighted sum of the values, which represents the output of the attention mechanism. By selectively attending to relevant information, attention mechanisms can improve the efficiency and robustness of the learning process [ZLC23].

Transformer architecture [VSP+23] was originally developed on attention mechanism for natural language processing (NLP) tasks, but their ability to model long-range dependencies and capture global context has made them attractive for perception tasks in autonomous driving. It aims to process and capture dependencies in input data, which eliminates the need for recurrent or convolutional layers, enabling highly parallelized computation. A typical Transformer consists of an encoder-decoder structure. The encoder is composed of a stack of identical layers, each containing two primary components: a multi-head self-attention mechanism and a position-wise feed-forward neural network. The multi-head attention module enables the model to simultaneously weigh the importance of different parts of the input sequence relative to each other, capturing long-range dependencies. Transformer architecture incorporates positional encoding, which injects information about the relative or absolute position of input elements, as the attention mechanism does not inherently capture positional information [ZLC23].

In autonomous driving applications, Transformer-based architectures have been widely adopted in a variety of sub-tasks, including object detection [LWL+22; LWZS22], lane detection [PCF+22], and segmentation [LWX+22], tracking and localization [NAZ+22; ZCW+22], path planning, and decision-making [HYC+23; ZTG+22]. Additionally, recent studies have explored the use of

Transformer in constructing end-to-end deep learning models [HYC+23] for autonomous driving. These models leverage the attention mechanism to further improve their ability to focus on relevant information and perform effectively in complex, real-world driving scenarios [ZLC23].

2.4 Multi-modal Feature Fusion

Simultaneous multimodal sensations are a crucial enabler of human perceptual learning [SG05]. For artificial learning systems, however, designing a unified model for modality fusion is challenging due to a number of factors: (i) variations in learning dynamics between modalities [WTF20], (ii) different noise topologies, with some modality streams containing more information for the task at hand than others, as well as (iii) specialised input representations [NYA+22].

The self-attention operation of transformers provides a natural mechanism to connect multimodal signals. Multimodal transformers have been applied to various tasks including audio enhancement [EML+18], speech recognition [HTG16], image segmentation [YRLW19] as well as autonomous driving [CPJ+22; WSDY23]. Wayformer [NAZ+22] used concatenation and attention encoder for feature fusion and discussed differences between early fusion, late fusion, and hierarchical fusion in the context of motion forecasting.

2.5 Scene Understanding

Scene understanding aims to recognize the semantic information of objects within their contextual environment. It is involved in various tasks like object detection, semantic segmentation, visual question answering, etc., and plays a critical role in robotics, autonomous driving, smart city, etc.. Significant advancements have been made by current supervised methods for 2D and 3D scene understanding[HZM+22; WLJ+22; WLL+22]. However, these methods heavily rely on extensively annotated datasets, which pose significant challenges when encountering unseen instances that were not included in the training data.

In contrast, the CLIP (Contrastive Language-Image Pre-training)[RKH+21] leverages contrastive learning, a powerful technique that enables the model to learn rich semantic representations from unlabeled data. By training on large-scale datasets with diverse image-text pairs, CLIP learns to associate semantically similar concepts while distinguishing between dissimilar ones. This enables CLIP to develop a nuanced understanding of scene semantics, allowing it to be used as a generalized scene semantic feature encoder in this work. Without leveraging the power of CLIP, we would need a large-scale dataset that includes various complex driving scenarios to train our semantic feature encoder, which is very costly to collect.

Initially designed for image classification, CLIP falls short in segmentation and localization performance[CLK+23]. These two kinds of tasks appear to be dramatically different: localization tasks are vision only and require fine-grained output (e.g., bounding boxes or pixel masks), while VL understanding tasks emphasize fusion between two modalities and require high-level semantic outputs (e.g., answers or captions) [ZZH+22]. Zhang et.al. proposed GLIPv2, a grounded VL understanding model as a unified model for localization and VL understanding tasks. They reformulate localization tasks as VL grounding tasks, in which the language input is a synthesized

sentence as the concatenation of category names [LZZ+22]. In GLIPv2, they introduce the novel inter-image region-word contrastive learning task, which leverages phrases from other sentences in the same batch as potential negatives, as another much stronger VL grounding task. This new region-word contrastive loss enables GLIPv2 to learn more discriminative region-word features and achieves mutual benefit between localization and VL understanding [ZZH+22].

3 Methodology

We have information from various sensors including cameras, lidar, radar, and GPS. Additionally, the high-definition map that is pre-generated by Bosch is available for the whole route in the dataset. The camera can give information about the target vehicle about the driver inside, brake light, type of vehicle, and surrounding scenes. Radar and Lidar can give information about the velocity and relative position of the target vehicle and nearby traffic participants. GPS and maps can give information about the type of road.

The information from various sensors included in our dataset should be sufficient for our neural network to distinguish vehicles with different motion statuses in most of the cases. In order to do that, we need to build special encoders for each modality for feature extraction and fuse those features into a single latent feature that can be used for classification. Details for this feature extraction and fusion pipeline will be discussed in the following sections.

3.1 Overall Architecture

Let us first take a look at the overall architecture of our feature extraction fusion pipeline as shown in Fig3.1 before diving into details for each component.

In the previous chapter, we proposed several questions that human drivers will think about when evaluating the motion status of a nearby vehicle. Inspired by that, we designed our model so that all the questions raised above can be answered by part of our model.

States encoder utilizes information from radar and lidar to answer questions about motions and locations of different traffic participants:

- Are there people next to the car?
- What are the former states (velocity, position) of the car?
- Are there obstacles / pedestrians in front of the car?

Global vision encoder utilizes information from the entire photo captured by the camera to address questions regarding the semantics information about the scene as well as the target vehicle:

- What kind of car is it?
- Is this a parking area?
- Is here a controlled-access road?

Object-level vision encoder utilizes information from the image crop around the target vehicle to answer visual questions specifically related to it:

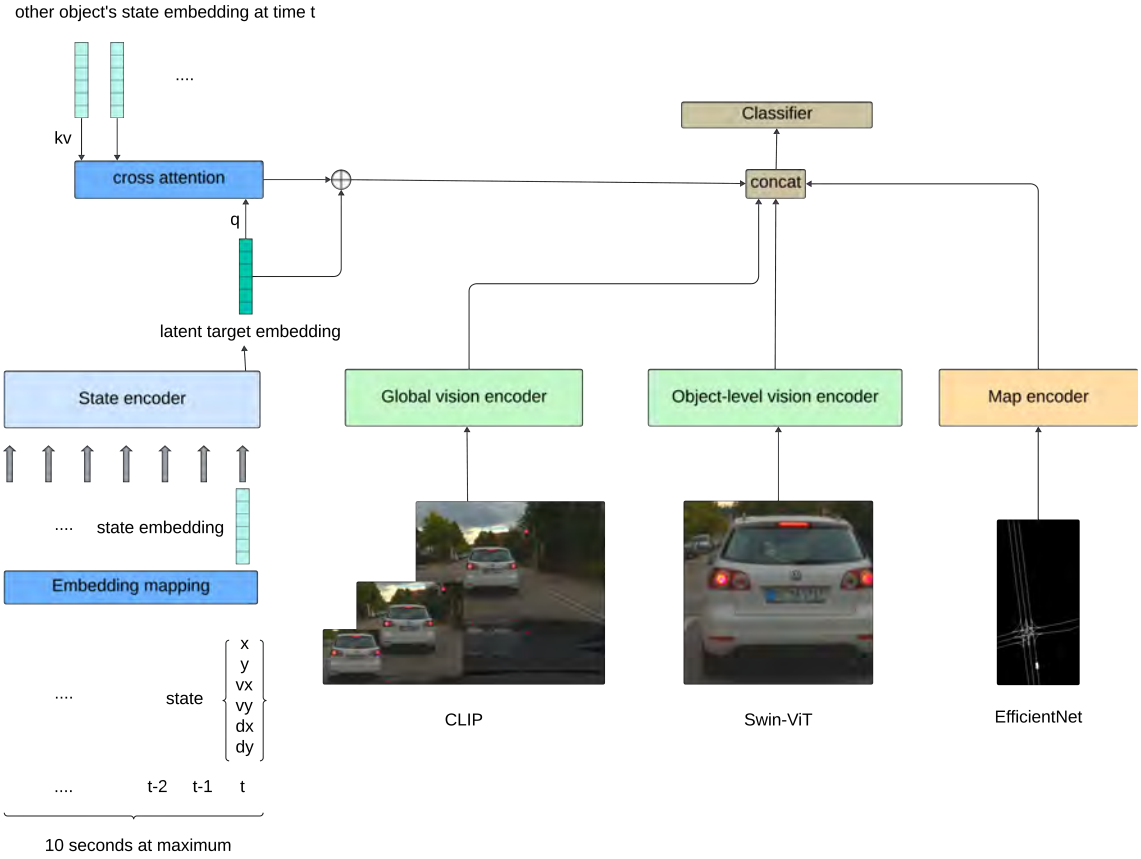


Figure 3.1: Architecture of our model. States encoder utilizes information from radar and lidar to output features representing the motion and locations of the target vehicle. Global and object-level vision encoders encode image crops to latent features representing the scene semantics and vehicle attributes. Map encoder encodes the map data to latent features representing the type of road and location of the target vehicle on the map

- Is there a driver inside?
- Are the brake lights on?
- What kind of car is it?

Map encoder utilizes the high-definition map and the target vehicle's position on it to answer questions about the location from a high-level perspective:

- Is this a parking area?
- Is here a controlled-access road?

In the following, we will call the four encoders mentioned above *latent encoder*, which stands for the encoder that encodes the part of the input data into latent information that serves a certain need.

With the latent features provided by the four latent encoders, we can then combine them by concatenating the four latent features into one fused feature. The fused features are then fed into the classifier which is a linear layer to output the logits for three categories.

In summary, the four latent encoders illustrated in Fig 3.1 are listed as follows:

- States encoder: Encoding the former tracked states of the target vehicle with a transform encoder. The last token of the output sequence is used as the latent target feature that represents the motion of the target vehicle. This latent target feature will then attend to nearby traffic participants' states through a cross-attention computation and a residual addition.
- Global vision encoder: Encoding the multi-scale image crops around the target vehicle with a pretrained CLIP model. CLIP features from multi-scale image crops are then aggregated into singe latent semantic feature that represents the semantic information of the scene related to the target vehicle.
- Object-level vision encoder: Encoding the image crop of the target vehicle for object-level attributes like type of vehicle, and brake light state with a pretrained Swin Transformer.
- Map encoder: Encoding the map data with EfficientNet. The latent feature represents the type of road and location of the target vehicle on the map.

In the following sections, the details about the structures of the four latent encoders and their inputs and outputs will be introduced.

3.2 States encoder

We define the *state* of a vehicle at time frame t as its velocity $v_t \in R^2$, position $p_t \in R^2$, and orientation $d_t \in R^2$. Those attributes of vehicles at different times is measured by lidar and radar and preprocessed with the sensor fusion pipeline of Bosch. Details on that will be discussed in Section 4.1. The aim of the states encoder is to encode the former states of the target vehicle.

Before entering the encoder, the *state* of all vehicles is first transformed into target-centric normalized coordinates and mapped into a *state embedding* of higher dimension. Specifically, the *state* of a vehicle at time t, $\{v_t, p_t, d_t\} \in R^6$ is mapped into $s_t \in R^d$ where d is the dimension of our latent feature space. As shown in Fig 3.2, the transformation is done in three steps.

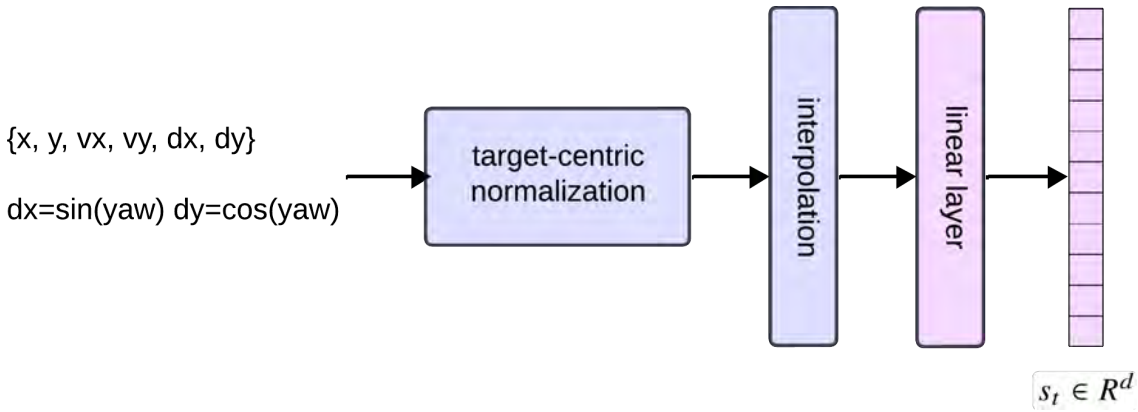


Figure 3.2: Overview of embedding mapping process that consists of three steps

For the first step, the state is transformed with respect to the position and orientation of the target at time t and then normalized to values between 0 and 1 through min-max normalization. The normalization is done on position and velocity separately, the orientation which is computed with sine and cosine functions is normalized in nature. The transformation makes the state value independent of the ego vehicle's motion and thus makes it easier for the model to capture the pattern.

For the second step, the former states of the target vehicle that will be used as the input of the states encoder are interpolated. Note that this step doesn't affect the state of nearby traffic participants because only the state at time t is considered in those cases. The interpolation is necessary because the tracked former states of the target vehicle are not always distributed uniformly on time. This could be a problem because the positional embeddings typically only represent the index of the input token instead of its time stamp. To fix this problem, the input states are interpolated with equal time step size through linear interpolation.

For the third step, the target-centric interpolated normalized state will be projected into a higher dimension d with single linear layer with bias and ReLU activation.

After the mapping process, we encode embeddings of the former states of the target vehicle into a latent target feature as shown in Fig 3.3. The input sequence of the transformer encoder is the sequence of former states embeddings mixed with positional embeddings. The last token of the output sequence is used as the latent target embedding representing the state of the target vehicle while attending to the tracked history of it.

While transformers typically utilize pooling layers or a special classification token (CLS) to summarize the features of the entire sequence [DBK+21; DCLT19], our approach takes the last layer hidden state of the input sequence's last token as the latent target embedding. It encapsulates the latent representations with an emphasis on the state at time t, which is the time frame where we make decision on.

In the transformer encoder, positional embedding is an additional embedding that is added to the input embeddings before they are processed by the model. They provide the model with information about the positions of tokens in the sequence, as transformers do not inherently possess built-in notions of order or position. In our case, the positional embeddings represent the time step for each state in the history.

While in the original paper [VSP+23] the positional embeddings are calculated using sine and cosine functions of different frequencies and phases. Remarkable follow-up works tend to use learnable 1D positional embeddings [DBK+21; DCLT19]. The learnable positional embeddings are learned along with the other parameters of the model during training. In this way, the model is endowed with the ability to adaptively learn representations of position that are tailored to the specific characteristics of the input data. Hence, we also use the learnable positional embeddings in this work.

It is very common for a vehicle to stop to give way to other vehicles or pedestrians. Instead of only considering the positions and motion of the target vehicle, it will be beneficial to also take the interaction between the target vehicle and nearby traffic participants into consideration.

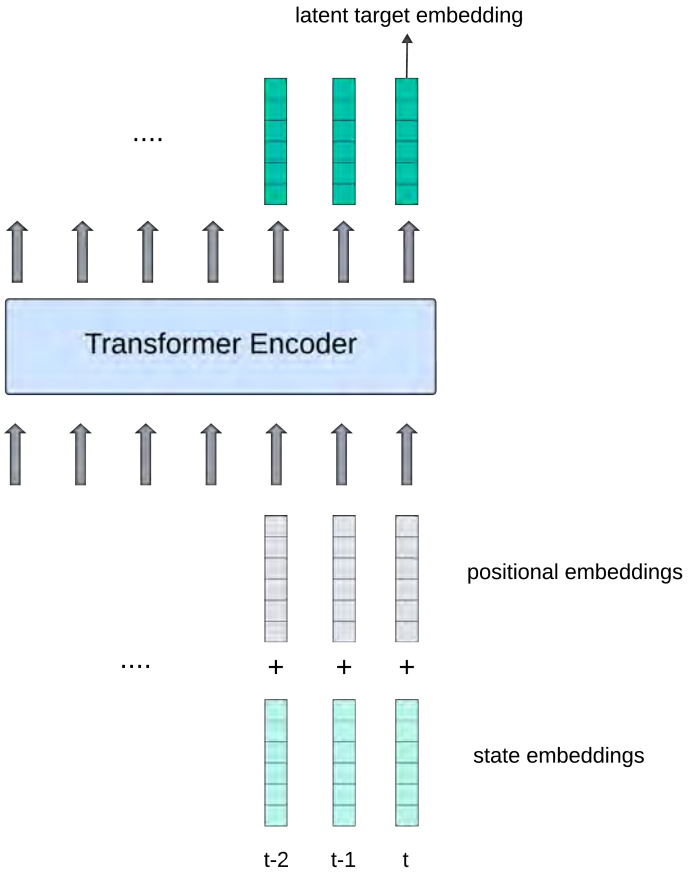


Figure 3.3: We use a transformer encoder as the states encoder for encoding embeddings of the former states of the target vehicle. The positional embeddings are learnable parameters random initialized. The last token of the output sequence is used as the latent target embedding.

Other than being used as a powerful feature extractor that can capture long-range dependencies and global context on the input, attention mechanism is commonly used for encoding the interactions between different agents in motion forecasting works [MYVN21; ZWLH23]. Schmidt et.al. have quantitatively shown that the self-attention mechanism is able to learn social interactions between vehicles [SJGD22].

As shown in Fig. 3.4, inspired by the social attention in motion forecasting works [MYVN21; ZWLH23], we use a cross-attention computation in which the latent feature of the target vehicle are used as query and the state embeddings of other nearby traffic participants at time t will be used as key and value. The output of this social encoder will be a latent feature that represents the relative relationship between the target and other agents and it will be added to the original latent target feature in a residual addition manner.

Note that our design differs from [MYVN21; ZWLH23] in that the former state embeddings of the neighbours here are not encoded by the states encoder like the target vehicle. There are several reasons behind this. First of all, unlike motion forecasting where the target's future trajectory is affected by the trajectory of its neighbours. The positions and velocities of the neighbours are

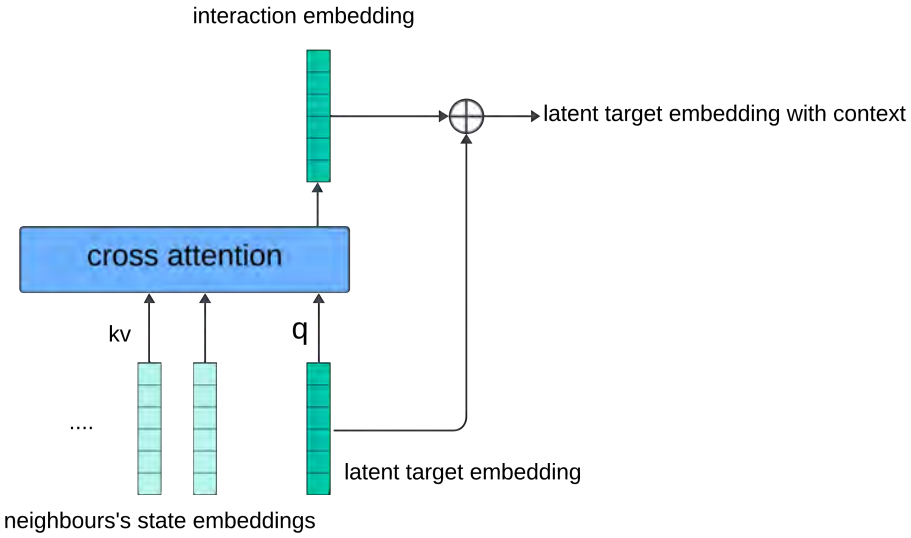


Figure 3.4: We use a cross attention mechanism to encode the interactions between the target vehicle and nearby vehicles and pedestrians (neighbours). The interaction embedding is then added with latent target embedding for the target embedding with context.

sufficient in our case to determine whether the target vehicle is blocked by other traffic participants. Second, encoding the former states of all the neighbours is a big computation overhead, especially considering that all these states need to be first transformed into the target vehicle's coordinates before entering the state encoder, which means that they can't be reused for different targets. This problem has been discussed in detail in [ZWLH23].

Due to this difference, the target embedding and neighbours' state embeddings share the same dimension but don't live in the same space. Hence, instead of a self-attention mechanism, we use a cross-attention mechanism to extract the interaction embedding and add it on top of the latent target embedding.

3.3 Global Vision Encoder

With the global vision encoder, we aim to encode the rich semantic features of the scene related to the target vehicle. As discussed in Section 2.5, CLIP's pretraining on a vast dataset comprising diverse visual concepts ensures that it can generalize well to novel environments and tasks, making it a versatile and powerful choice for this. However, CLIP embeddings are global in nature and not suitable for target-focused feature extraction. That is to say, the CLIP model is trained to output embeddings that represent the whole image without being able to align it to a certain pixel or area, as shown in 3.5. This can be a problem in our application because we only want the semantic feature of the scene surrounding the target vehicle instead of the whole scene that may contain a lot of irrelevant objects like the houses on the side of the road.

There are many works[KKG+23; ZSZ+24] that aim to ground CLIP embeddings into 3D fields at pixel level for 3D scene understanding. In order to do pixel-aligned feature extraction, they use methods based on CLIP embeddings across multi-scale crops of training views. Inspired by this, we

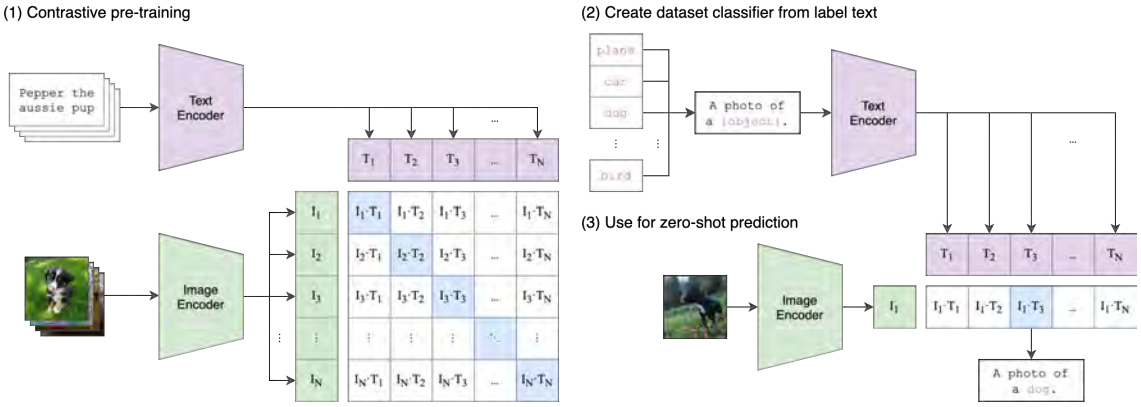


Figure 3.5: CLIP overview. From Radford et al.'s *Learning Transferable Visual Models From Natural Language Supervision*[RKH+21]

adopt a similar method where CLIP embeddings are computed on multi-scale crops centered at the target vehicle as shown in Fig 3.6. These crops' embeddings are then fused with a 1D convolution layer.

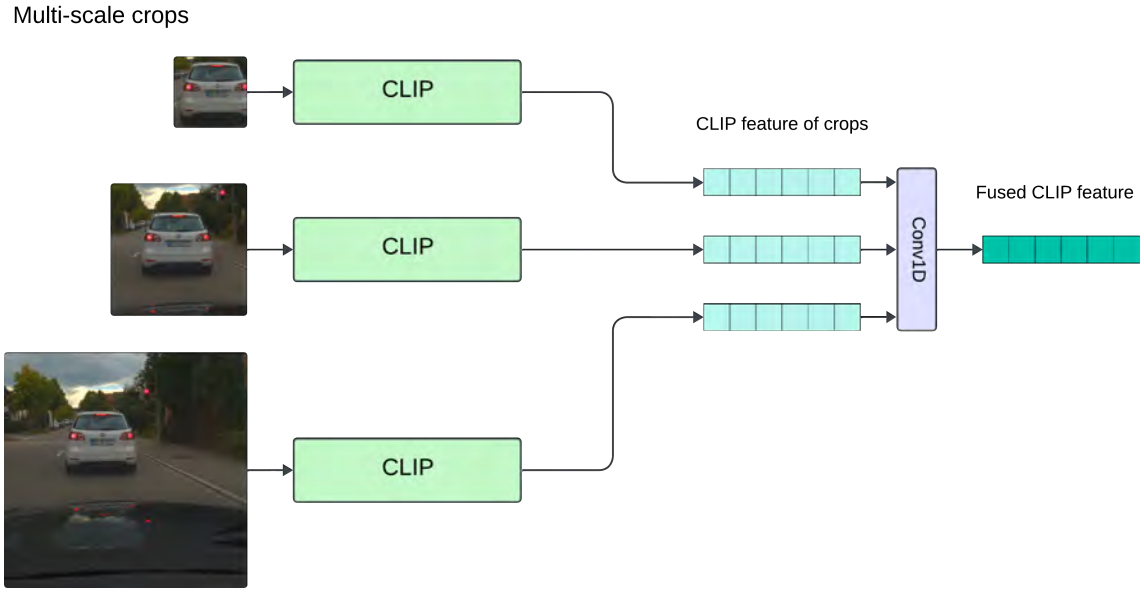


Figure 3.6: CLIP features are computed on multi-scale crops centered at the target vehicle and then fused by a 1D convolution layer

As discussed in Section 2.5, GLIPv2[ZZH+22] serves both localization tasks that require fine-grained output and vision-language understanding tasks that require high-level semantic outputs by unifying localization pre-training and vision-language pre-training with three pre-training tasks. Using its pre-trained encoder as our encoder might result in a performance gain because in our application we also expect the vision encoder to capture features at different levels. However, due to that the code of GLIPv2 is not yet released, we leave this experiment for the follow-up work.

3.4 Object-level Vision Encoder

While being a powerful vision encoder, CLIP's pretraining focuses on learning to associate images with text across a wide range of concepts, enabling it to perform tasks such as image-text retrieval and zero-shot classification rather than fine-grained object detection. Hence, CLIP can capture high-level semantic features of objects, but it does not provide the level of granularity needed for tasks such as identifying brake light states or vehicle types. Therefore, while CLIP is a powerful tool for understanding visual content at a global level, it is not directly suitable for serving as an object-level encoder without additional modifications or adaptations.

In this context, training a specific Swin Transformer [LLC+21] at the object-level emerges as a good choice. By feeding the model with cropped images centered at the target vehicle as shown in Fig 3.1, we expect the model to extract specific visual features such as brake lights and vehicle types. This approach enables the vision transformer to focus on the object of interest within the image, leveraging its capabilities to capture fine-grained details and object-specific attributes.

Capturing attributes like brake light blinking ideally requires an encoder designed for video data, which can effectively analyze temporal dynamics and sequential patterns. Despite numerous works focusing on video understanding, encoding an image sequence with transformers entails significant computational overhead. This complexity poses challenges for real-time inference, especially in applications such as autonomous driving where low latency is critical. Thus, while using a video encoder as our object-level encoder might have a small positive impact on the overall performance, we choose to use an image encoder due to its significantly lower computational burden relative to the minor benefits a video encoder offers.

3.5 Map Encoder

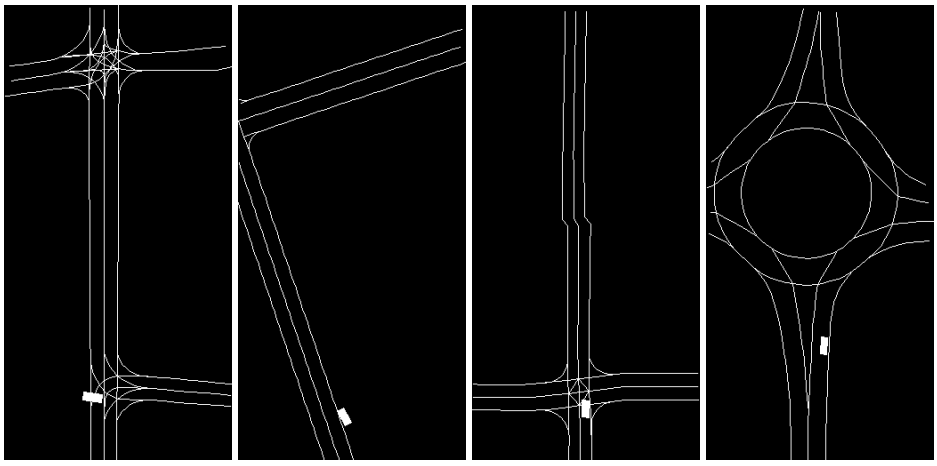


Figure 3.7: Map examples

The high-definition map is generated by Bosch and aligned with the ego vehicle's location through the self-localization component which is part of Bosch's sensor fusion pipeline. The map data comes as a list of line segments that can be drawn to a binary image as shown in Fig 3.7. The drawn

map is relative to the ego vehicle's location and orientation. The location of the ego vehicle (not drawn on the map) on the map is at the bottom center with the head vertically upward. Other than the line segments, we also annotate the location of the target vehicle with its relative pose to the ego vehicle and its width and length estimated by lidar / radar.

Instead of representing the map as a binary image, it is a common practice in motion forecasting works[NAZ+22; WSDY23; ZWLH23] to represent road graph segments as polylines, approximating the road shape with collections of line segments specified by their endpoints and annotated with type information. However, in our application, the directions of the road are not important to us. We only care about where the target vehicle lies on the map, so there's no need to transform every segment of the roadgraph into vector space.

Due to the simplicity of the map image structure and the trivial feature we want to extract from it, we decided to use a convolutional neural network (CNN)-based model due to its better computationally efficiency compared to transformer-based models. EfficientNet[TL20] emerges as a good choice due to its good performance while maintaining efficiency in terms of model size and computational cost. Map images are cropped from thebottom to fit the input size of EfficientNet.

4 Dataset

Existing large-scale datasets[GLSU13; WQA+23] for autonomous driving don't have annotations for stopped vehicles. To fill this gap, we collect the first dataset that includes labels for the vehicle's motion status(parked, stopped, moving) in dense traffic scenarios that include various cases of stopped vehicles. The dataset consists of 7 data sequences. Each of them were taken from a dense traffic route between the Bosch Renningen Campus and the city center of Renningen. It takes on average 25 minutes for a human driver to drive through this route and the data from sensors is recorded along the way. The data-collecting vehicle is equipped with high-resolution cameras, lidar, radar, accelerometer, and GPS.

The transformation from the raw recordings to the dataset for training proposes several challenges. The raw dataset contains a lot of outliers, ghost objects, and objects that are too far away. The identification of stopped vehicles is complex so careful manual labeling is required. Another challenge is that the dataset is very unbalanced because most of the vehicles we encounter are either parked or moving instead of stopped. The details on how to tackle these challenges are discussed in the following sections.

4.1 Data Pre-processing

The recorded sensor data are processed by a sensor fusion pipeline of Bosch where the objects including vehicles and pedestrians are detected and tracked. For each time frame, we get an object list containing all the objects that are tracked in this frame. Each element in this list contains various information about the object and also an object ID that is supposed to be identical for the same object across the whole sequence. From the object list, we extract the velocity, position and orientation of the tracked objects. For each time frame, we also extract the photo of size 2560×1216 taken by the front camera at that time and the map image for each object as shown in Fig 3.7.

As shown in Fig 4.1, typically the front view photo contains a lot of objects. In order to obtain image crops for use in the global and object-level vision encoders, we must first align the position and size information derived from lidar and radar in 3D space with a bounding box on the 2D front view photograph. With the front camera's intrinsic and extrinsic parameters, this is done by composing the perspective projection matrix.

We use the pinhole camera model to describe the mathematical relationship between the coordinates of a point in 3D space and its projection onto the image plane of our front-view camera. According to it, the projection relationship is described by:

Figure 4.1: An example of the photo from the front camera

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} \tag{4.1}$$

Where u and v are the x and y coordinates of the pixel in the camera, x_e, y_e, z_e are the coordinates of the source of the light ray which hits the camera sensor in ego vehicle coordinates. For the same object, this ego vehicle coordinate is assumed to be the same when captured by the camera and detected by the lidar/radar at the same time. z_c is the z-coordinate of the camera relative to the world origin, which is divided from the matrix product for the pixel coordinates. R, T are the extrinsic parameters which denote the coordinate system transformations from 3D ego vehicle coordinates to 3D camera coordinates where R is the rotation matrix and T is the translation matrix. R, T is determined by the relative position between the ego vehicle and the front view camera and is obtained by the camera calibration conducted by Bosch before. K is the intrinsic matrix as described below:

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where α_x and α_y represent focal length in terms of pixels, γ represents the skew coefficient between the x and the y axis and u_0 and v_0 represent the principal point, which would be ideally in the center of the image. These parameters are associated with the front view camera and is obtained by the camera calibration conducted by Bosch before.

With the target vehicle's size $\{l, w, h\}$ where l, w, h is the length, width, and height of the vehicle, we can obtain the coordinates of 8 corners of the 3D bounding box in target vehicle coordinates. With the target vehicle's position $\{x_e, y_e, z_e\}$ and orientation in ego vehicle coordinates, we can transform those corner coordinates into the ego coordinates. After that, with equation 4.1, we can project those corner coordinates into pixels on the front view image. By taking the minimum and maximum of these 8 corners in the x and y axes, we can reduce the number of corners to 4 and have the approximate bounding box in 2D space. With the 2D bounding box, we can extract image crops around the target vehicle on the front view image.

4.2 Data Filtering & Labeling

When we have the processed data, the next step is to label the data and filter it. First of all, for each sequence, we filter out a list of vehicles that are at least once close enough to the ego vehicle. The detection range of lidar and radar is very far compared to the camera. Since we use the camera input as an important part of our model, we must limit the predictable vehicles to the vehicles that can be seen by the front camera. As shown in Fig 4.2, only the vehicles that enter the prediction area will be included in the predictable vehicles list. The prediction area is thinner horizontally because too many irrelevant vehicles will be included if the horizontal distance is 15 meters, e.g. vehicles behind a fence or a wall. When a vehicle first enters the prediction area, its ID is included in the predictable vehicles list as long as the time stamp which will be used as the prediction time.

Based on the predictable vehicles list, we generate videos with a frame rate of 10 from the front view image sequence with object ID, bounding box, and velocity drawn for each predictable vehicle with OpenCV. An example of a frame of the annotated video is shown in Fig. 4.3. On the upper left corner there is the time stamp of the current frame. There are three vehicles tracked on this frame, each of them is surrounded by red dots that represent their bounding box. Their object ID, and the vertical and horizontal velocity with respect to the ego vehicle are shown above the bounding box.

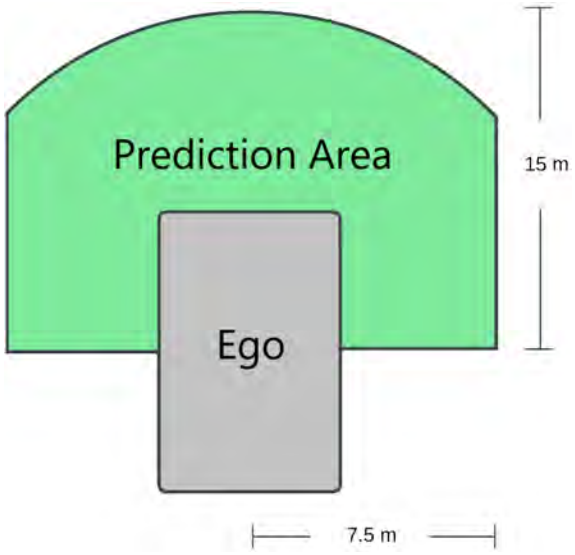


Figure 4.2: Prediction area. Only vehicles that enter the prediction area is considered

Figure 4.3: One frame of the annotated video

Annotated video simplifies the labeling process for predictable vehicles, as we can easily identify and label them by going through the video and observing the annotations. Each vehicle requires labeling only once, as predictions are made based on their motion status upon entering the prediction area for the first time, even if the motion status of some vehicle changes (e.g., transitioning from moving to stopped). The prediction time is automatically determined when a vehicle is included in the predictable vehicles list. This method reduces the need to record prediction times via timestamps. Instead, we only need to record the object ID and label associated with it. This simplifies the labeling process very much. For special cases where the prediction time is inappropriate, e.g. the vehicle can't be seen or can only partly be seen by the front camera, we will manually adjust the prediction time.

After the labeling, we need to filter the dataset again because the sensor data is noisy sometimes and there are many outliers in the dataset. For all the vehicles that are labeled as "parked", the norm of the difference of their positions in the tracked history should not be larger than 1. For vehicles that are labeled as "moving", the norm of the maximum velocity in the tracked history should not be smaller than 1. The vehicles that fail to meet these criteria will be removed from the dataset. Other than that, vehicles that have a tracked history that is too short (shorter than 0.5 seconds) will also be removed because their motion status is hard to tell in this case.

Number of sequences	7
Number of parked vehicles	856
Number of stopped vehicles	66
Number of moving vehicles	383
Number of vehicles in total	1305

Table 4.1: Overview of our dataset after labeling and filtering

4.3 Data Augmentation

As shown in Table 4.1, the dataset is very unbalanced because it is much easier to encounter parked or moving vehicles than stopped vehicles. Before data augmentation, there were 856 parked vehicles and 383 moving vehicles while the number of stopped vehicles is only 66. This will make it very hard for our model to generalize well on different labels.

In order to solve this problem, instead of one prediction time we use a short sequence of time as prediction time and treat the same stopped vehicle at different time frames as independent samples during training and testing as shown in Fig 4.4. This is done by manually recording the suitable prediction time range for each stopped vehicle during the labeling process. A suitable prediction range here means that for each time step within this range, the target vehicle should already have a long enough tracked history and the motion status of the target vehicle within this range should remain stable. Although these instances are similar to each other each of them still carries special features. The model should learn by trying to discover the commonality of them. Note that the data samples in the training set and test set are independent of each other. After the data augmentation, we now have 819 stopped instances derived from 66 independent vehicles, which is already comparable to the number of parked vehicles.

During the training process, we utilize the *WeightedRandomSampler* from Pytorch and adjust the weight for each instance based on their label so that for each batch that is drawn for training, the labels for the instances are uniformly distributed. For the test set, in order to keep the accuracy a fair metric for all labels, we assign the same number of instances for each label in the test set, which is 169. The split of training and test dataset is done randomly for each category once.

	Training set	Test set
Number of sequences	7	
Number of parked vehicles	687	169
Number of stopped vehicles	650	169
Number of moving vehicles	214	169
Number of vehicles in total	2058	

Table 4.2: Overview of our dataset after data augmentation and training/test set split

Figure 4.4: Instances derived from the same stopped vehicle

5 Experiments

5.1 Implementation Details

The dimension d of state embeddings and latent features as stated in the previous chapters are 768. For the global vision encoder, we use a pre-trained CLIP model with base-sized ViT as its vision encoder and a batch size of 32. The dimension of the CLIP output is 512 which is then projected into d. The object-level vision encoder is a tiny-sized Swin Transformer pre-trained on ImageNet-1k at resolution 224x224. The output of the Swin Transformer is 768. The map encoder is an untrained EfficientNet B0 model. The reason why we don't use a pre-trained model here is that the map images in our application differ a lot from the natural images in the pre-training dataset like ImageNet-1k. The states encoder is implemented following the original transformer paper[VSP+23] and the number of transformer blocks is one. The projection layer and the final classifier are both single linear layers. Dropout layers are placed after projection layers and output layers in states encoder for regularization. In the end, our model has a size of 207M parameters.

For training, we use a cross entrop loss and a SGD optimizer with a momentum of 0.9, a learning rate of 0.0001, and weight decay of 0.02. All models discussed and compared in this chapter are trained with early stopping for best test performance. The training is done with a NVIDIA GeForce RTX 2070 SUPER with a batch size of 16.

Before the joint training of the full model. We first pretrain the map encoder by only feeding latent features from the map encoder to the classifier. The idea is that in this way we give the map encoder a warm-up to help it extract meaningful information and make it comparable to other pretrained encoders. This can prevent the model from greedily relying on other modalities and keeping the map encoder underfit. Of course, the accuracy will be very low when only relying on map data. We will early stop the training process before overfitting the map encoder. More details on this pretraining process and the reason behind it can be found in Section 5.4.4.

We don't use methods introduced in previous works [BMBL19; YLLS21] discussed in Section 2.1 as our baseline for two reasons. First of all, these two works aim at the binary classification of parked / moving vehicles and "should overtake" / "shouldn't overtake" and select crafted features based on their goal and dataset. This makes their method not applicable to our task because we aim at three categories classification on another dataset where the stopped vehicles are very hard to identify based on the features they selected. Second, the features they crafted based on their dataset aren't present in our dataset. As a result, we use a single states encoder and our classifier layer as the baseline model where no information about the scene is included.

5.2 Results

In this section, we will present and analyze the results of our full model.

5.2.1 Metrics

As discussed in Section 4.3. Our test dataset consists of 169 instances for each category and we evaluate our model on it with accuracy, recall and confusion matrix. Other than that, we include two fully independent sequences (no instance in these sequences is in the training dataset) as an additional assessment for the generality of our model on unseen data.

5.2.2 Performance

Metrics	Baseline	Ours
Accuracy	83.23%	97.63%
Recall Parked	98.82%	97.63%
Recall Stopped	52.66%	98.22%
Recall Moving	98.22%	97.04%

Table 5.1: Accuracy and recalls of our model and baseline on test dataset. All models are trained with early stopping for the highest test accuracy and the same hyperparameters.

		Predicted Labels		
		Parked	Stopped	Moving
True Labels	Parked	165	4	0
	Stopped	1	166	2
	Moving	0	5	164

Table 5.2: Confusion matrix of our model

		Predicted Labels		
		Parked	Stopped	Moving
True Labels	Parked	167	2	0
	Stopped	74	89	6
	Moving	1	2	166

Table 5.3: Confusion matrix of baseline

We first compare the performance of our model with the baseline on our dataset. As shown in Table 5.1, 5.2, and 5.3, the baseline fails to distinguish parked and stopped vehicles as we expected while our model achieves 97.96% accuracy with a 98.22% recall on stopped vehicles. This shows that our model does succeed on this three categories classification task with the help of multimodal input and scene understanding.

From the confusion matrix, we can observe that the differentiation between parked and moving vehicles is easy for our model while sometimes it confuses parked and moving vehicles with stopped vehicles. This could be due to the noise in our dataset. The measurements of the sensors are not always stable during the real driving situation, especially when it comes to the estimation of the ego vehicle's motion. The estimation of other vehicle's position and velocity is based on the estimation of the ego vehicle's motion, so they are not always accurate. It is very common that a parked vehicle has a small velocity in our dataset.

From Table 5.4 we can observe that parked vehicles that are wrongly classified as stopped vehicles have larger average velocity at detection time. On the other hand, moving vehicles that are wrongly classified as stopped vehicles have much lower average velocity. From this, we can tell that some of the errors may be introduced by the detection noise of the parked vehicle and some of the slowly moving vehicles.

		Predicted Labels		
True Labels	v_t m/s	Parked	Stopped	Moving
	Parked	0.11	0.25	/
	Moving	/	3.22	11.58

Table 5.4: v_t here stands for the magnitude of velocity at detection time t. In this table, we group the instances by their true labels and predicted labels and show the average v_t for each group.

Then we evaluate our model with two independent short sequences that are recorded from the same route with the same set of sensors on different days. Each of them has a length of 8 minutes and none of their data is included in the training dataset. Thus, we can prevent if there is any information leaking across different samples in the same sequence.

From Table 5.5, 5.6, and 5.7 we can see that there's some performance drop on the recall of parked vehicles in sequence 0. Other than that, our model remains very high accuracy on independent sequences.

Metrics	Sequence0	Sequence1
Accuracy	91.58%	97.81%
Recall Parked	80.49%	95.45%
Recall Stopped	100%	98.65%
Recall Moving	100%	100%

Table 5.5: Accuracy and recalls of our model on two independent short sequences

		Predicted Labels		
True Labels		Parked	Stopped	Moving
	Parked	33	8	0
	Stopped	0	37	0
	Moving	0	0	17

Table 5.6: Confusion matrix of our model on independent sequence 0

True Labels \ Predicted Labels	Parked	Stopped	Moving
Parked	42	2	0
Stopped	1	73	6
Moving	0	0	19

Table 5.7: Confusion matrix of our model on independent sequence 1

5.3 Model Interpretation

In this section, we will have a deeper look into what our model is paying attention to and try to add some explainabilities to our model's decision process.

5.3.1 Impact of Latent Encoders

First of all, let us see which latent encoder is contributing more to the result. As shown in Figure 5.1, we use a single linear layer as our classification head and it takes the concatenated latent features from four latent encoders as input. This means that each of these latent features contributes to the output logits for certain categories ("parked" in the figure) independently. This is helpful for us because the parameters in the classification head can directly be used as a measurement for the impact of each latent encoder.

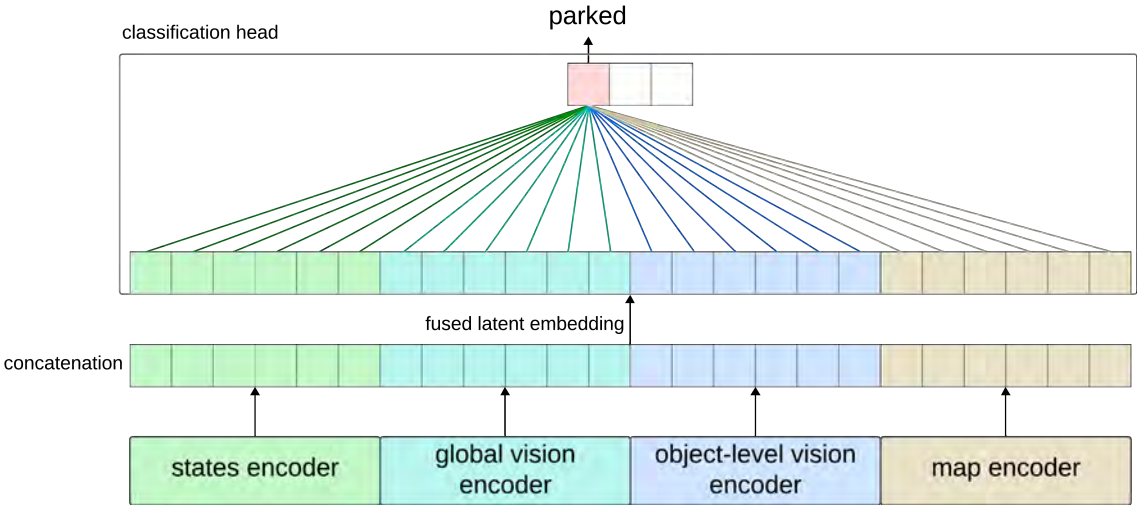


Figure 5.1: The latent features from four latent encoders contribute to the output logits of each category independently

In formal, given the parameters of the classification head $P_{cls} \in R^{4d \times 3}$, the impact I_i^c $(i < 4)$ of the ith latent encoder for category c $(c < 3)$ is described as:

$$I_i^c = \frac{mean(abs(P_{cls}[id : (i+1)d, c]))}{\sum_{c' \neq c} mean(abs(P_{cls}[id : (i+1)d, c']))}, I_i^0 + I_i^1 + I_i^2 = 1 \tag{5.1}$$

Where $mean$ and abs stand for the average and absolute value function. By using this equation, we can get a table as shown in Table 5.8. States encoders have the highest impact on the decision for moving vehicles while object-level vision encoder has the highest impact on the decision for parked and stopped vehicles. This makes sense because the best way of distinguishing a moving vehicle is to see if its position is changing over time and if it has a high velocity. On the other hand, it is hard to distinguish parked from stopped vehicles relying on the states encoder because both parked and stopped vehicles are not moving. Hence, it makes sense that the model relies more on the object-level vision encoder in this case because looking at the vehicle directly is the best way of telling whether the vehicle is started (looking at the brake lights and the driver inside).

	states encoder	global vision encoder	object-level vision encoder	map encoder
parked	0.24	0.25	0.28*	0.23
stopped	0.24	0.24	0.28*	0.24
moving	0.35*	0.22	0.22	0.22

Table 5.8: Impact of latent encoders on the output. All the values are rounded to two decimal places. Highest impact value for each category is marked with *

5.3.2 Visualization of Object-level Vision Encoder

Following the analyses above, we further investigate what our vision encoders are paying attention to. With the great advancements that vision transformers bring to the computer vision field. Researchers also made great efforts in interpreting them [AZ20; CGW21]. However, most of the research in this field focused on ViT. Swin Transformer uses attention matrixes of different sizes and are shifted during the computation, which makes its attention matrixes hard to visualize. Famous existing methods [AZ20; CGW21] are not applicable to Swin Transformer. Hence, in the following, we will visualize the attention of our object-level encoder which is a Swin Transformer with transformers-interpret[1] library that is based on Captum[2] and uses the integrated gradients method for the Interpretation. The integrated gradients method computes the integral of the gradients of the model's output with respect to its input along a straight path from a baseline input (usually an input with zero relevance) to the actual input. It assigns attribution scores to input features based on how much they contribute to the model's output change.

As shown in Fig 5.2, we can observe that our object-level vision encoder learns to pay more attention to the brake lights in order to determine that this vehicle is stopped. Interestingly, when we visualize the attributions to label "parked", the brake lights area becomes negative impact (annotated in blue instead of red). This confirms our assumption on the reason for the high impact of object-level vision encoder on prediction on labels "parked" and "stopped". The object-level vision encoder distinguishs them through the brake lights states. Most of the time stopped vehicles have their brake lights on or blinking.

[1] https://github.com/cdpierse/transformers-interpret
[2] https://github.com/pytorch/captum

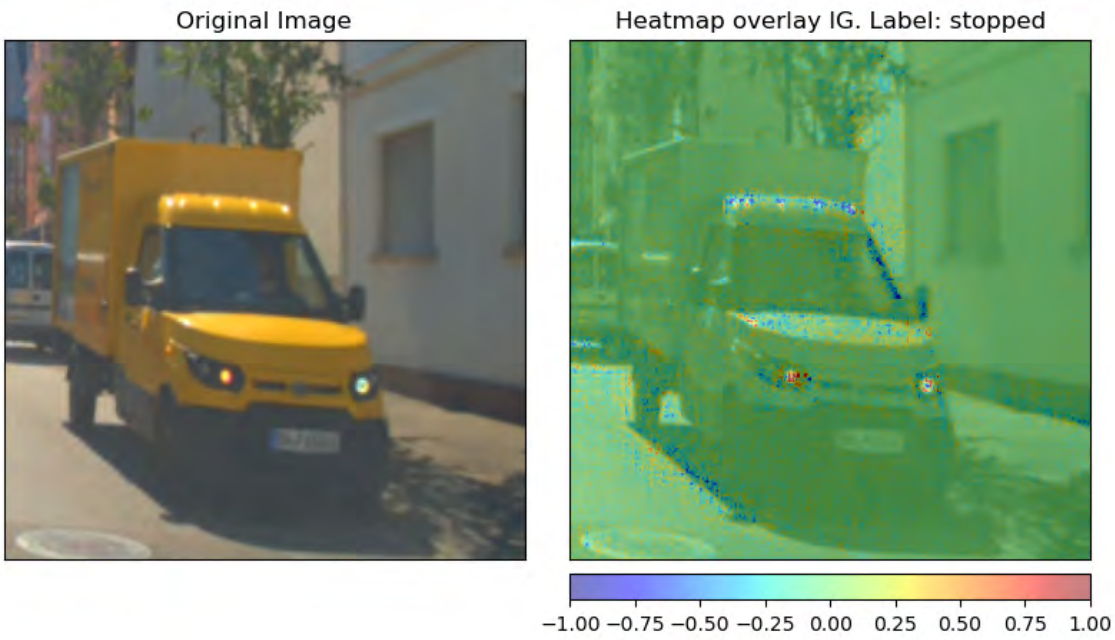


Figure 5.2: On the left: A stopped van. On the right: The attributions to label “stopped” of every pixel in the image through the object-level vision encoder (Swin Transformer) in our trained model.

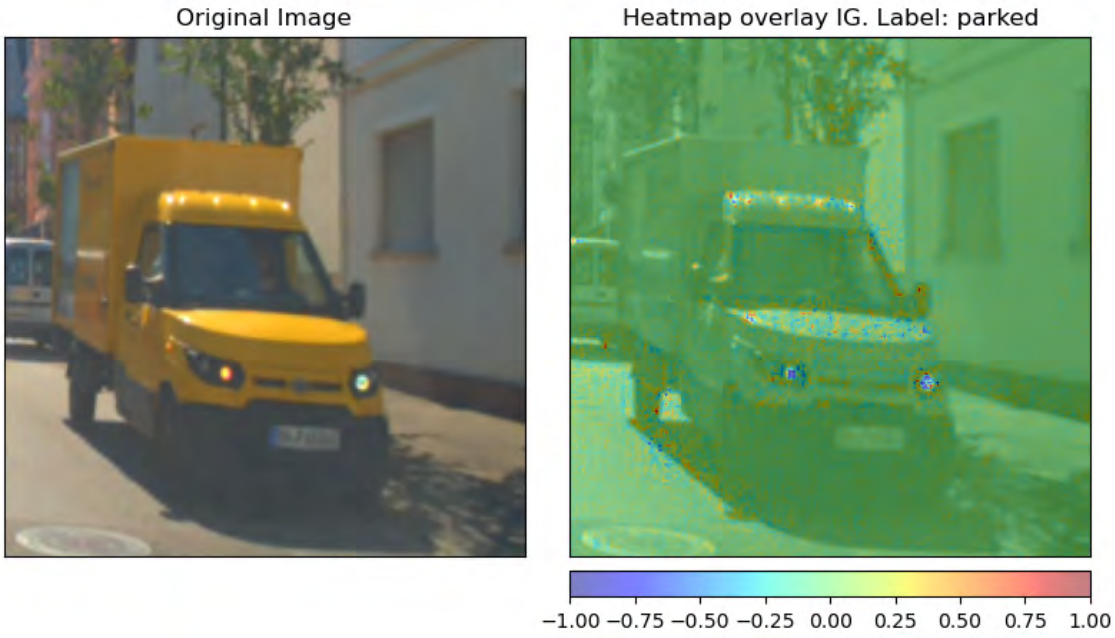
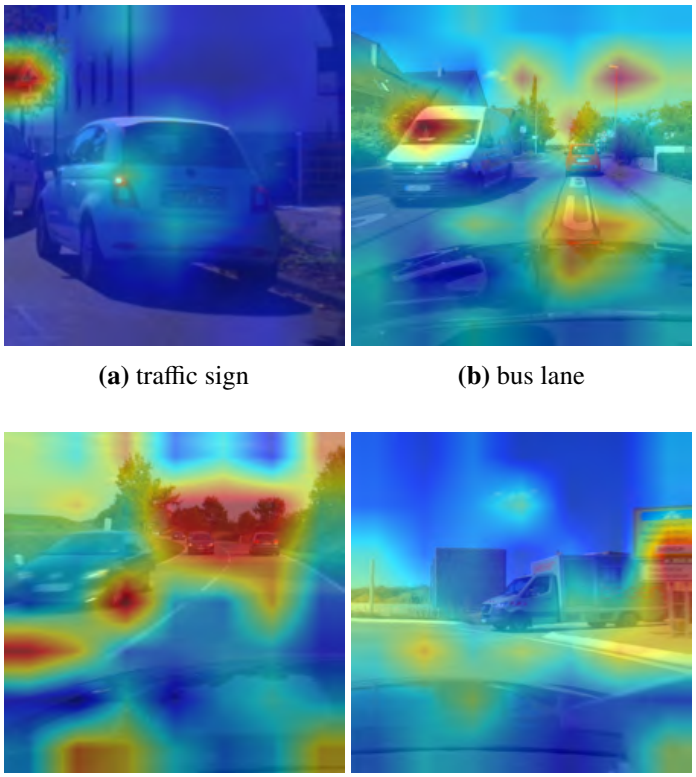


Figure 5.3: On the left: A stopped van. On the right: The attributions to label “parked” of every pixel in the image through the object-level vision encoder (Swin Transformer) in our trained model.

5.3.3 Visualization of Global Vision Encoder



(a) traffic sign **(b)** bus lane

(c) neighbouring vehicles **(d)** traffic sign

Figure 5.4: Good examples of fine-tuned CLIP paying attention to meaningful areas like traffic signs, bus lane, neighbouring vehicles

In this section, we will try to add some explainability to our global vision encoder, the fine-tuned CLIP model. CLIP uses a base-sized ViT as its vision encoder. As discussed in the previous section, there have been many works on the Interpretation of ViT and we use attention rollout [AZ20] in this work.

At every transformer block we get an attention matrix A that defines how much attention is going to flow from token j in the previous layer to token i in the next layer with element A_{ij}.

We can multiply the matrices between every two layers, to get the total attention flow between them. Taking the residual connections into consideration, we can recursively compute the attention rollout matrix at layer l:

$$AttentionRollout_l = (A_l + I)AttentionRollout_{l-1} \tag{5.2}$$

where I is the identity matrix. $AttentionRollout_L$ for the deepest layer will be the attention matrix that can be used to show the attention flow from each patch to the final representation. Note that here the attention flow isn't related to certain output labels because we are mainly interested in knowing the fine-tuned CLIP model's attention distributions in general.

From Fig 5.4 we can observe that the fine-tuned CLIP model is capable of paying attention to meaningful semantic information that is related to our task. For example, traffic signs, bus lanes, neighbouring vehicles, and the road. From Fig 5.5 we can observe that the fine-tuned CLIP model shows different attention patterns at different scales. At low scale, it pays more attention to the target vehicle. At medium scale, it starts to focus on nearby objects like traffic cones and fences. At high scale, the model tends to pay more attention to the road and nearby vehicles.

However, we find out that the attention of the fine-tuned CLIP model is very easy to be distracted by irrelevant objects. As shown in Fig 5.6, trees are especially easy to distract the fine-tuned CLIP model. Other irrelevant things like parts of the ego vehicle, houses by the street, sky, etc. are also frequently the focus of attention. This problem arises from the paradox of our expectation of the CLIP model as the global vision encoder. On one hand, we expect CLIP to retain its generality so that it can extract semantic information from objects that are unseen or rarely seen in the training dataset. On the other hand, we don't want it to focus on irrelevant objects like trees and thus introduce noise to the model.

A potential solution to this problem is to retrain a foundation model that is specially tailored for autonomous driving problems using contrastive learning by limiting the images and texts to traffic-related field. This is beyond the scope of this work and we hope future works will fill this gap.

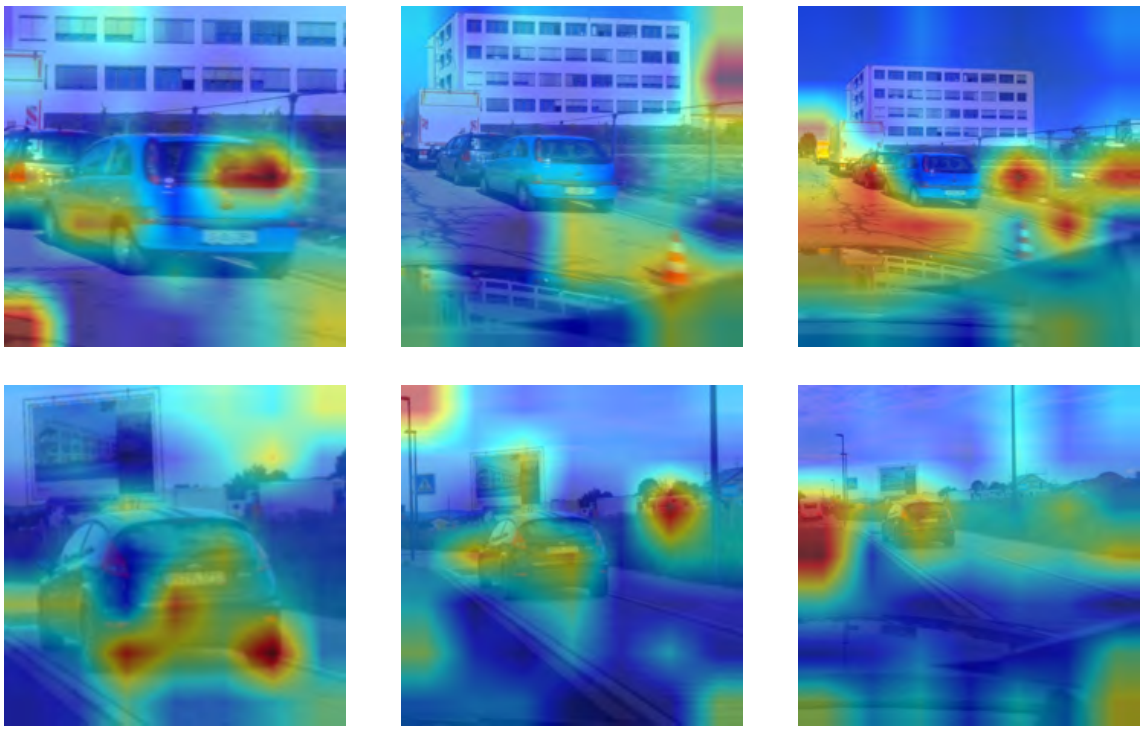


Figure 5.5: Examples of fine-tuned CLIP paying attention to different things at different scales

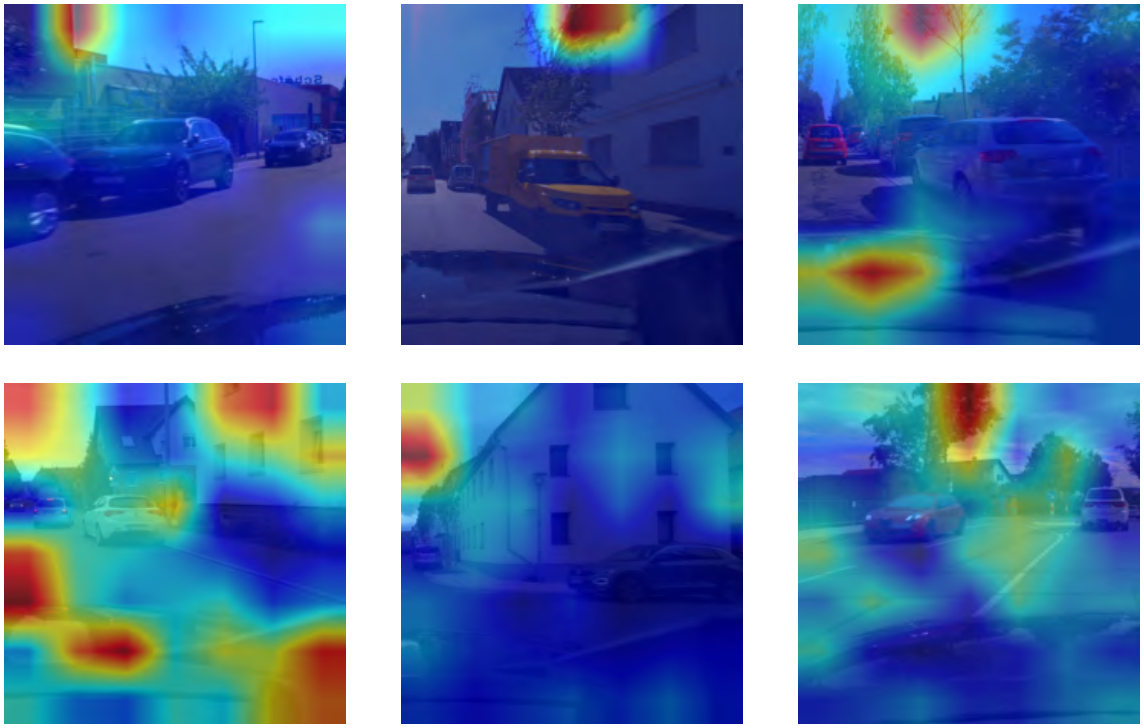


Figure 5.6: Examples of fine-tuned CLIP get distracted by irrelevant objects

5.3.4 Visualization of Map Encoder

In this section we would like to investigate whether our map encoder is paying attention to meaningful features. To visualize the EfficientNet that we use as the map encoder, we use Grad-CAM [SCD+19], a well-known method that is typically used for the visualization of convolution neural networks. Grad-CAM leverages the gradient information flowing into the last convolutional layer of the network to compute the importance of each feature map. By weighting the feature maps based on their importance, it produces a heatmap that indicates the regions most relevant to the network's prediction.

We first focus on cases where we need the map encoder the most. As shown on the left of Fig 5.7, these three stopped vehicles are hard to be distinguished from parked vehicles by the latent features from the states encoder and object-level vision encoder. The states encoder has difficulty distinguishing parked and stopped vehicles because, as discussed before, both stopped and parked vehicles are stationary. The object-level vision encoders also have difficulties in these cases because the brake lights are not on or visible. On the other hand, map information can be useful in this situation because stopped vehicles typically appear at the intersection of the roads or in the middle of a multi-lane road where vehicles are not normally allowed to be parked. Hence, the position of the target vehicle over the map can be strong evidence to a stopped vehicle.

As shown on the right of Fig 5.7, high relevance is shown between the output of our trained map encoder and the class "stopped". The map encoder learns to detect that the vehicle is at the intersection or in the middle of the road and relates that to the class "stopped".

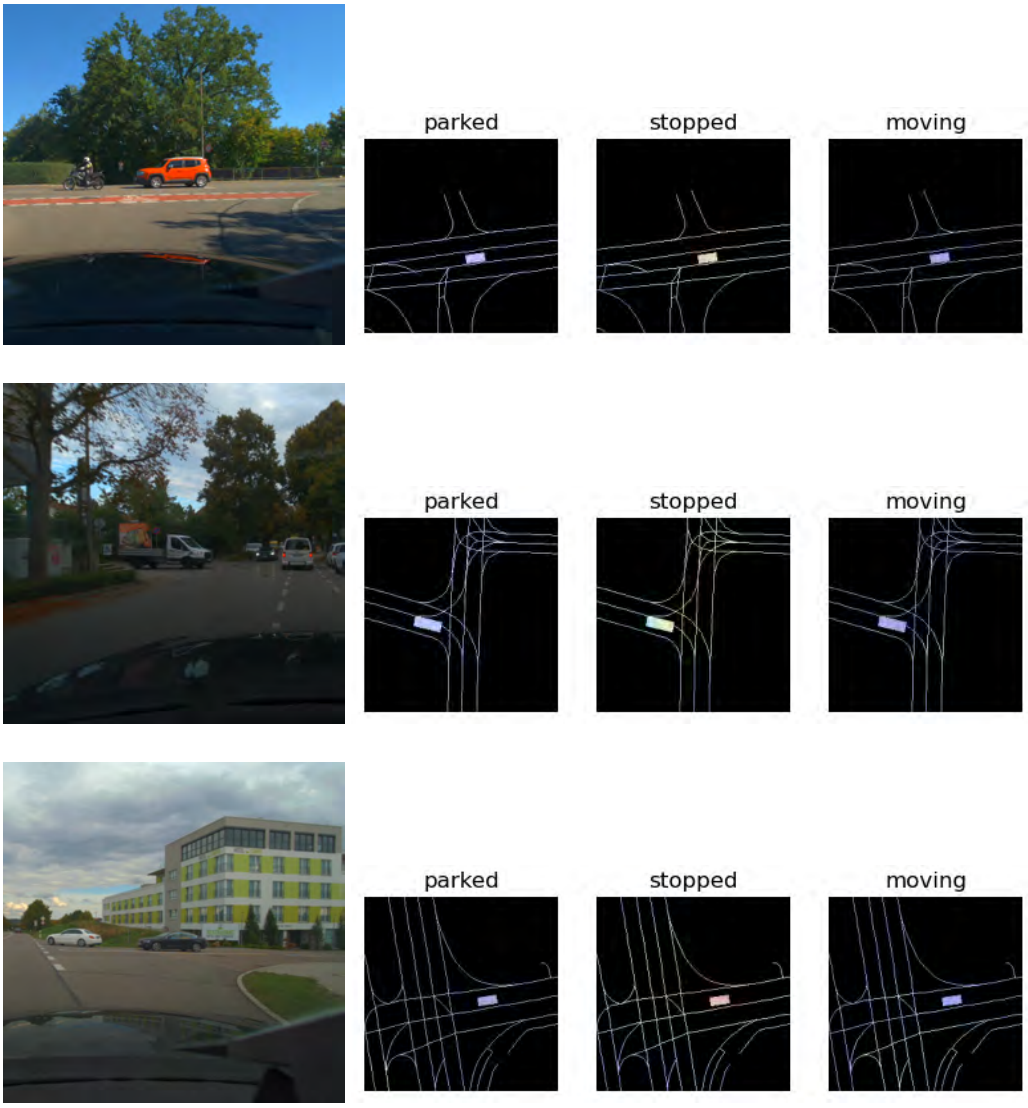


Figure 5.7: On the left: Original images of stopped vehicles that are hard to be distinguished from parked vehicles through lidar / radar or camera. On the left: Importance heat map of map encoder, generated with Grad-CAM. Reddish areas mean high relevance to the output logit corresponding to the target class shown above. A larger relevant area also means a larger output logit and a higher possibility of the target class.

5.4 Ablation Studies

In this section we will introduce the ablation studies performed on our model and explain some of our design decisions based on the experiment results.

5.4.1 Feature Fusion

As discussed in the Section 2.4, the self-attention operation is commonly used for feature fusion due to its ability to capture long-range dependencies. For simplicity, we simply use concatenation for the fusion of four latent features and use a single linear layer as the classification head. This design doesn't allow the latent features to communicate with each other. Instead, they "vote" for the final results independently as shown in Fig 5.1. In this section, we investigate whether fusing the latent features through self-attention will have a positive impact. Instead of directly concatenating the four latent features, we first pass them into a self-attention operation to let them communicate with each other and then concatenate them.

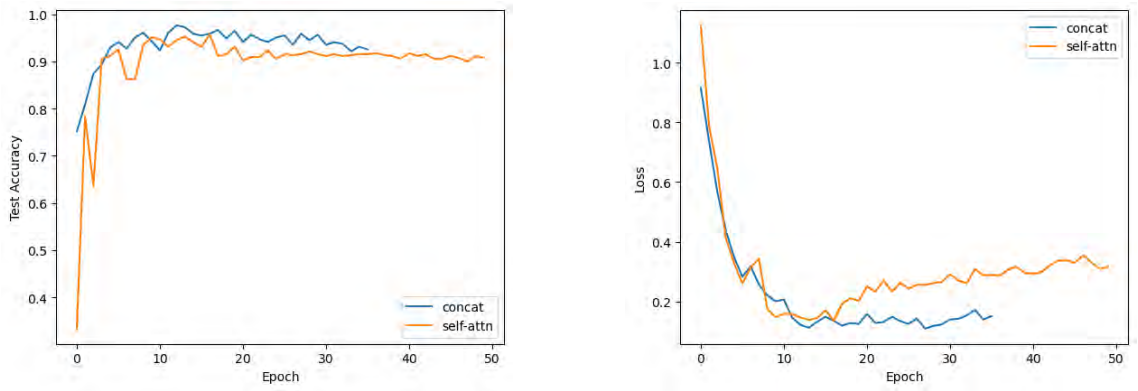


Figure 5.8: On the left: Comparison of test accuracy between our model using concatenation as latent features' fusion method and the model using self-attention instead. On the right: Comparison of test loss between the two models.

As shown in Fig 5.8, using self-attention as the method for latent features fusion results in slightly lower peak test accuracy (2.56% decrease) and comparable test loss. Self-attention operation will introduce more parameters into the model, without obvious performance gain, using concatenation as the method for latent features' fusion is sufficient for our model on our dataset. However, it is possible that on larger and more complex datasets allowing latent features to communicate will result in better performance.

5.4.2 States Encoder Pooling Method

As discussed in Section 3.2, we take the last layer hidden state of the input sequence's last token as the latent target embedding because it encapsulates the latent representations with an emphasis on the state at decision time t. In this section, we investigate whether appending a classification token (CLS) at the beginning of the states and use it as an aggregation over the former states results in better performance.

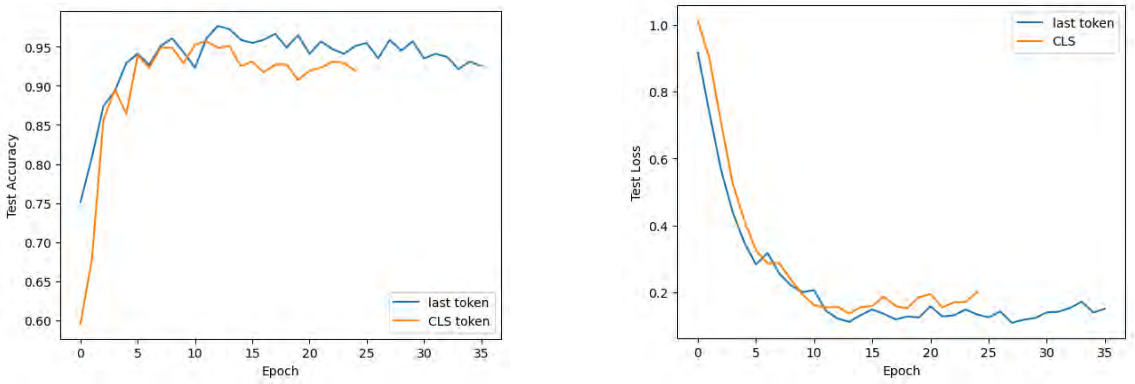


Figure 5.9: On the left: Comparison of test accuracy between our model using the last token as the latent target embedding and the model using the CLS token instead. On the right: Comparison of test loss between the two models.

As shown in Fig 5.9, there will be small peak test accuracy degradation and test loss increase if we use the CLS token as the latent target embedding that summarizes the former states of the target vehicle.

To get a better understanding of the differences here, we take a closer look at the accuracy, recall and the latent encoders' impact. From Fig 5.11, we can observe a drop of 2% in the impact of the states encoder on the moving vehicles prediction. This is a negative reflection on the quality of the states encoder. The model learns to rely more on other encoders when the states encoder is not as reliable as before. As shown in Table 5.9, although there is only a small accuracy degradation overall and a small increase in the recall of the parked vehicles, the recall of stopped vehicles drops by 6.5%.

Metrics	CLS	Last token
Accuracy	95.86%	97.63%
Recall Parked	98.82%	97.63%
Recall Stopped	91.72%	98.22%
Recall Moving	97.04%	97.04%

Table 5.9: Comparison of test accuracy and recalls between our model using the last token as the latent target embedding and the model using the CLS token instead. All models are trained with early stopping for the highest test accuracy and the same hyperparameters.

		Predicted Labels		
		Parked	Stopped	Moving
True Labels	Parked	167	2	0
	Stopped	14	155	0
	Moving	0	5	164

Table 5.10: Confusion matrix of the model using the CLS token as the latent target embedding

	states encoder	global vision encoder	object-level vision encoder	map encoder
parked	0.24	0.25	0.28*	0.23
stopped	0.24	0.24	0.28*	0.24
moving	0.33*	0.22	0.24	0.22

Table 5.11: Impact of latent encoders of the model using CLS token as the latent target embedding. All the values are rounded to two decimal places. The highest impact value for each category is marked with *

5.4.3 Interpolation

In this section we investigate the usefulness of the interpolation in the former states of the target vehicle.

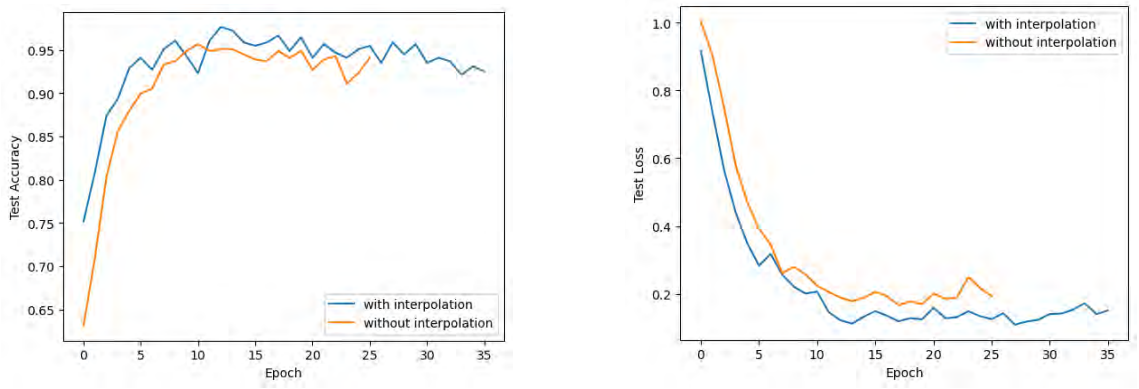


Figure 5.10: On the left: Comparison of test accuracy of our model with and without interpolation in the former states target. On the right: Comparison of test loss between these two situations.

As shown in Fig 5.10, applying interpolation on the states sequence will result in slightly higher peak test accuracy (1.97% increase) and lower test loss. Since little computation overhead is cost by the interpolation process, we consider it as a useful preprocess step.

5.4.4 Map Encoder Pretraining

Previous studies [WJCG22] on multi-modal deep neural networks have shown that models tend to rely on some modalities while under-fitting the other modalities. In our case, this phenomenon is especially likely to happen because we use pre-trained CLIP model and Swin Transformer as our vision encoders while the EfficientNet we use as the map encoder is not trained due to the big domain change. At the beginning of joint training, pretrained CLIP model and Swin Transformer immediately extract useful information while the map encoder is generating random output. The model will learn to give more weight to other encoders and thus let map encoder underfit. To prove our assumption on this, we conducted the following experiments.

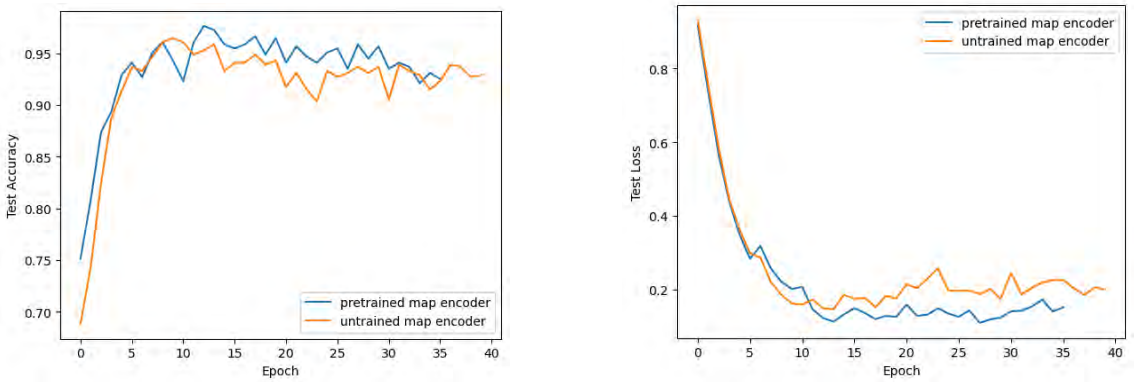


Figure 5.11: On the left: Comparison of test accuracy of our model with pretrained and untrained map encoder at the beginning of joint training. On the right: Comparison of test loss between these two situations.

From Fig 5.11 we can observe that the model with the map encoder pretrained has slightly higher peak test accuracy and lower test loss. However, from Table 5.14, we can observe that the impact of the map encoder drops on both parked and stopped vehicles prediction. This makes the map encoder the most unimportant latent encoder among the four latent encoders. As a result, we can observe from Table 5.12 and 5.13 that there is an obvious accuracy degradation in the prediction of parked vehicles because, as discussed before, information from the map provides an important clue for distinguishing stopped vehicles from parked vehicles.

In Fig 5.12 we compare the importance heatmap generated from the map encoder that is not pretrained before joint training with the heatmaps from Fig 5.7. It is obvious that without pretraining, the map encoder shows underfitting and gets less concentrated on its attention and decision.

Metrics	Map encoder w/o pretraining	Map encoder pretrained
Accuracy	96.25%	97.63%
Recall Parked	92.89%	97.63%
Recall Stopped	98.22%	98.22%
Recall Moving	97.04%	97.04%

Table 5.12: Comparison of test accuracy and recalls of our model with pretrained and untrained map encoder at the beginning of joint training. All models are trained with early stopping for the highest test accuracy and the same hyperparameters.

		Predicted Labels		
		Parked	Stopped	Moving
True Labels	Parked	157	12	0
	Stopped	1	166	0
	Moving	0	4	165

Table 5.13: Confusion matrix of the model using the untrained map encoder at the beginning of joint training

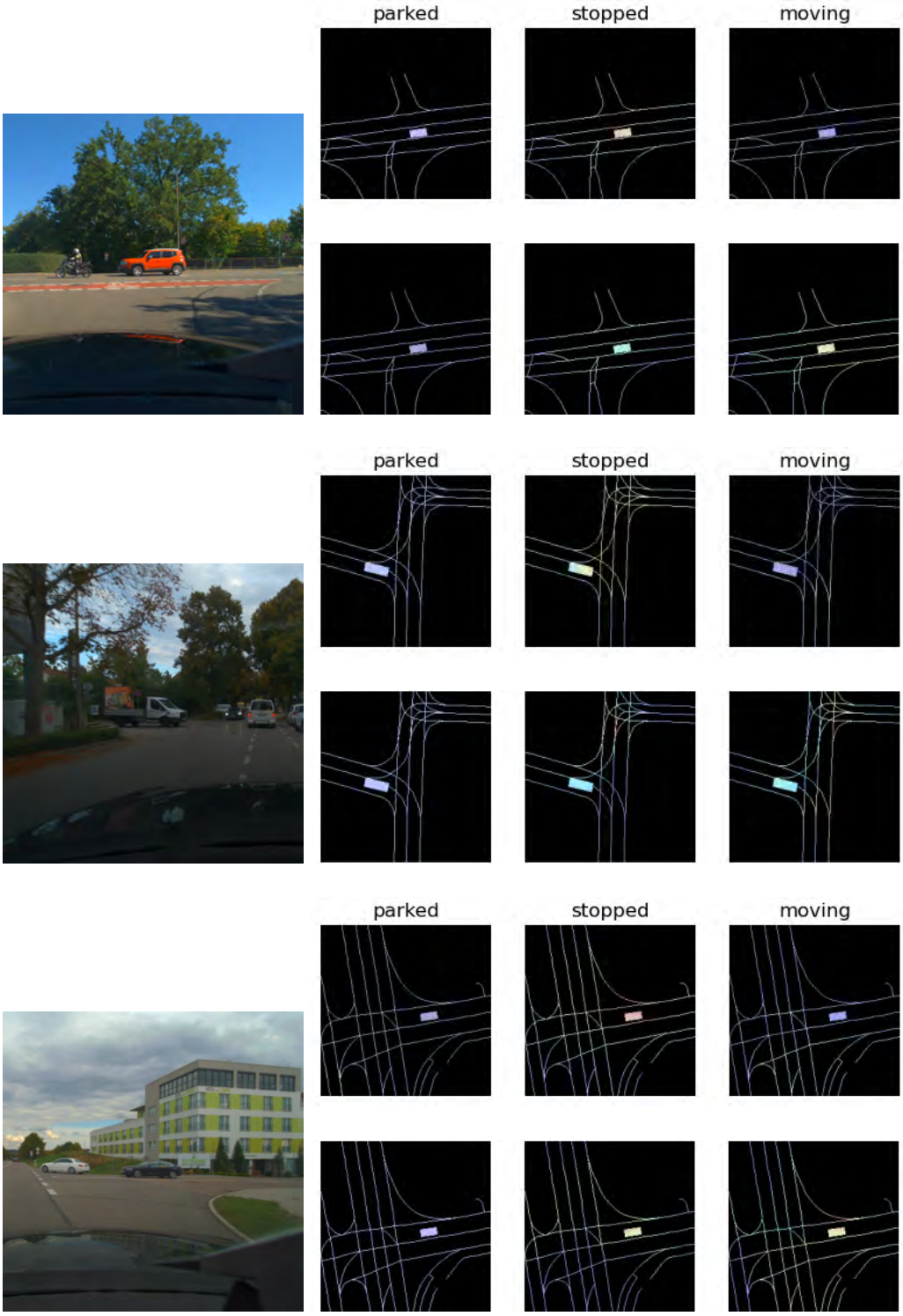


Figure 5.12: On the left: Original images of stopped vehicles that are hard to be distinguished from parked vehicles through lidar / radar or camera. On the left: Importance heat map of the map encoder. The heatmap above comes from map encoder that is pretrained before joint training as in Fig 5.7 while the heatmap below comes from map encoder that is not pretrained.

	states encoder	global vision encoder	object-level vision encoder	map encoder
parked	0.25	0.26	0.28*	0.22
stopped	0.25	0.25	0.28*	0.22
moving	0.34*	0.22	0.24	0.22

Table 5.14: Impact of latent encoders of the model using untrained map encoder at the beginning of joint training. All the values are rounded to two decimal places. The highest impact value for each category is marked with *

In conclusion, while there won't be a big performance degradation overall even if we don't pretrain the map encoder, we still believe that this is a meaningful step because it prevents the map encoder from being underfitted. We suggest that the small performance difference between these two setups is due to the limited size and complexity of our dataset. On a larger and more complex dataset, the differences could be bigger.

5.4.5 CLIP Features Fusion Method

As shown in Fig 3.6, we used a 1d convolution layer across CLIP representations at different scales which is basically a weighted average. In this section, we investigate whether another aggregation method will result in better performance. We evaluate two additional aggregation methods: average and self-attention operation.

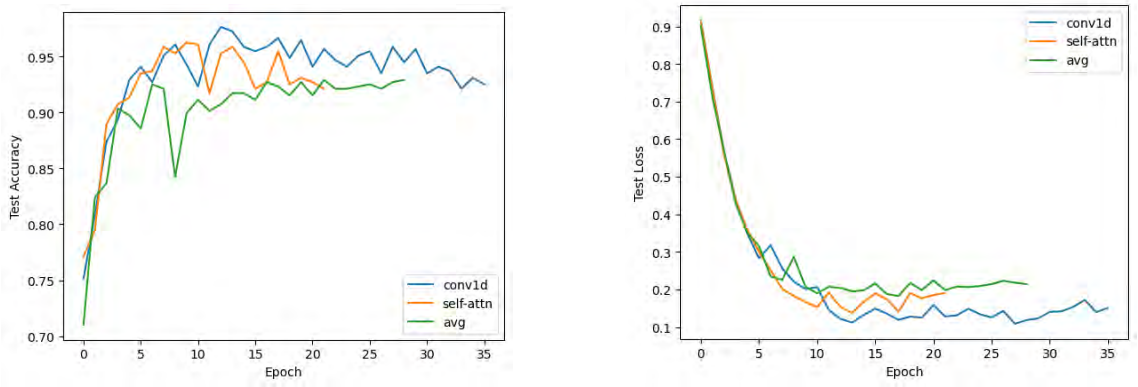


Figure 5.13: On the left: Comparison of test accuracy between three CLIP fusion approaches: 1d convolution, average and self-attention. On the right: Comparison of test loss between these three situations.

As shown in Fig 5.13, directly taking the average of multi-scale CLIP features as latent semantic features will cause an obvious degradation in both test accuracy (5.13% decrease) and loss. Using self-attention and pooling for aggregation will cause slightly lower peak test accuracy (1.38% decrease) and higher test loss. Also, the test performance is very unstable in this case. More complex architecture makes the model prone to overfitting.

In conclusion, the 1d convolution fusion gives the best performance without adding more complexity to the model.

5.4.6 Latent Encoder Selection

	Accuracy	Recall Parked	Recall Stopped	Recall Moving
full model	97.63%	97.63%	98.88%	97.04%
w/o map	95.26%	89.34%*	99.41%	97.04%
w/o global vision	94.08%	90.53%	95.86%	95.86%
w/o object vision	92.11%	89.94%	90.53%	95.86%
w/o states encoder	89.15%	95.26%	87.57%*	84.61%*

Table 5.15: Comparison of test accuracy and recalls between our full model and models without certain latent encoders. All models are trained with early stopping for the highest test accuracy and the same hyperparameters. "Map" stands for map encoder, "global vision" stands for the global vision encoder, "object vision" stands for object-level vision encoder. Lowest recalls are marked with *

Predicted Labels

w/o map encoder

True Labels	Parked	Stopped	Moving
Parked	151	18	0
Stopped	0	168	1
Moving	0	5	164

w/o global vision encoder

True Labels	Parked	Stopped	Moving
Parked	153	16	0
Stopped	7	162	0
Moving	0	7	162

w/o object-level vision encoder

True Labels	Parked	Stopped	Moving
Parked	152	17	0
Stopped	14	153	2
Moving	0	7	162

w/o states encoder

True Labels	Parked	Stopped	Moving
Parked	161	6	2
Stopped	12	148	9
Moving	7	19	143

Table 5.16: Comparison of test accuracy and recalls between our full model and models without certain latent encoders. All models are trained with early stopping for the highest test accuracy and the same hyperparameters.

In this section, we investigate the contributions of the four latent encoders by removing each of them from the model and evaluating the influences.

From Table 5.15, we can observe that in general removing the states encoder and object-level encoder will cause obvious performance degradation. This aligns with our intuition because these two encoders are responsible for fine-grained information (position, velocity, brake lights state) that is highly related to the target. Specifically, removing the object-level encoder will cause an obvious accuracy degradation in the recall of stopped vehicles. By looking at Table 5.16 we can find that this recall drop comes from wrongly classifying stopped vehicles as parked vehicles. This also aligns with our analyses before. The object-level encoder plays an important role in distinguishing stopped and parked vehicles by providing information about the brake lights.

The recall of moving vehicles remains high when removing map, global vision and object-level vision encoders. However, when states encoder is removed, the recall of moving vehicles drops to 84.61%. From Table 5.16, we can observe that for the first time, the model confuses parked vehicles

with moving vehicles. This also aligns well with our intuition and previous analyses. The states encoder is very important for identifying moving vehicles because it encoders the velocity of the target vehicle. The recall of stopped vehicles also drops by large because the model also starts to confuse stopped vehicles with moving vehicles without knowing the velocities.

Removing the map encoder and global vision encoder will not have a big influences except for the obvious accuracy degradation in parked vehicles. From Table 5.16, we can observe that the recall drop comes from wrongly classifying parked vehicles as stopped vehicles. In particular, a recall drop presents for all three labels when removing the global vison encoder. This suggests that the CLIP model still contributes to the identification of vehicles in all three categories despite being distracted by irrelevant objects.

Interestingly, removing the states encoder will have a very small influence on the recall of parked vehicles while removing any of the other latent encoders will cause an obvious drop. In summary, this shows that all the other latent encoders (map, global vision and object-level vision encoders) are contributing together to distinguish parked and stopped vehicles, while states encoder didn't provide useful information.

6 Conclusion and Discussions

In this thesis, we focused on the task of classifying vehicles into parked, stopped and moving vehicles. We motivate the need for vehicle classification by analyzing the challenge of the current autonomous driving systems in dense traffic and how vehicle classification can help interaction-aware prediction and planning. Then, we introduce a new category "stopped vehicle" to the previous vehicle classification problem for its potential benefit for the decision-making system. We continued by analyzing the limitations of existing works on vehicle classification and reasoning the necessary information that should be included for this problem. By analyzing the decision making process of human drivers on this problem, we correspond the necessary information to the measurements of sensors and propose a multi-modal model. To meet the complexity of our task and the needs of our model, we collect the dataset in real driving scenarios and then preprocess and label it. In the end, we implement and train this model and conduct thorough analyses and interpretations of it.

Our main contribution is that we design a novel multi-modal model that can leverage information from various sensors including lidar, radar, camera, GPS and high-definition maps for the task of vehicle classification. Four latent encoders that are responsible for extracting different kinds of features are included in our model. By introducing the CLIP model as our global vision encoder, we want to give our model the ability of scene understanding in order to generalize well on different driving scenarios and handle complex cases.

We investigate the effectiveness of each latent encoder through visualization methods and quantitative analysis. We find out that the model learns to combine the latent features from different encoders and the four latent encoders learn to solve a part of the problem with the information that they have. The Swin Transformer learns to extract information about brake lights states in order to distinguish stopped and moving vehicles from parked vehicles. The map encoder learns to detect whether the vehicle is at an intersection or in the middle of the road to distinguish stopped vehicles from parked vehicles. The states encoder learns to identify moving vehicles through velocity and position change. Removing one of these encoders will result in performance degradation in a certain part of the problem.

We find out that the fine-tuned CLIP model contributes to the identification of all labels based on the quantitative analysis. On the other hand, by visualizing the attention map of our fine-tuned CLIP model, we showed that directly applying foundation vision in the field of autonomous driving is not the best solution. The CLIP model is easily distracted by irrelevant objects and thus introduces noise to the model. More delicate fine-tuning and transfer learning could be done for better performance.

During the training process, we find out that directly applying the joint training of pretrained encoders and untrained encoders will cause the untrained encoders to be underfitted. To solve this problem, we pretrained our map encoder before joint training.

In this work, we simply use concatenation for the fusion of four latent features. We showed through experiments that a more sophisticated feature fusion method is not necessary in our case. However, our design does not allow communications between different latent features and this could limit its performance on larger and more complex datasets. In our model, the latent features essentially contribute through the classification head independently to the output. However, as shown in our ablation studies, different latent features don't always have useful information to provide for the identification of a certain vehicle. The uncertainty in this case could potentially contribute negatively, how to combine the information from different latent encoders wisely and only let certain latent encoders to contribute when they are certain about their decision is a very interesting direction for the follow-up work.

Bibliography

[AZ20] S. Abnar, W. Zuidema. *Quantifying Attention Flow in Transformers*. 2020. arXiv: 2005.00928 [cs.LG] (cit. on pp. 33, 35).

[BCB16] D. Bahdanau, K. Cho, Y. Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL] (cit. on p. 10).

[BKM+20] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, L. Eckstein. "The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1929–1934. DOI: 10.1109/IV47402.2020.9304839 (cit. on p. 7).

[BMBL19] K. Behrendt, O. Mangin, N. Bhakta, S. Lefevre. "Is this car going to move? Parked car classification for automated vehicles". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 541–548 (cit. on pp. 9, 29).

[CGW21] H. Chefer, S. Gur, L. Wolf. *Transformer Interpretability Beyond Attention Visualization*. 2021. arXiv: 2012.09838 [cs.CV] (cit. on p. 33).

[CLK+23] R. Chen, Y. Liu, L. Kong, N. Chen, X. Zhu, Y. Ma, T. Liu, W. Wang. *Towards Label-free Scene Understanding by Vision Foundation Models*. 2023. arXiv: 2306.03899 [cs.CV] (cit. on p. 11).

[CPJ+22] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, A. Geiger. *TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving*. 2022. arXiv: 2205.15997 [cs.CV] (cit. on p. 11).

[DBK+21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV] (cit. on p. 16).

[DCLT19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL] (cit. on p. 16).

[EML+18] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, M. Rubinstein. "Looking to listen at the cocktail party: a speaker-independent audio-visual model for speech separation". In: *ACM Transactions on Graphics* 37.4 (July 2018), pp. 1–11. ISSN: 1557-7368. DOI: 10.1145/3197517.3201357. URL: http://dx.doi.org/10.1145/3197517.3201357 (cit. on p. 11).

[GLSU13] A. Geiger, P. Lenz, C. Stiller, R. Urtasun. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013) (cit. on pp. 8, 23).

[HLW+23] Z. Huang, H. Liu, J. Wu, W. Huang, C. Lv. *Learning Interaction-aware Motion Prediction Model for Decision-making in Autonomous Driving*. 2023. arXiv: 2302.03939 [cs.RO] (cit. on p. 7).

[HTG16] D. Harwath, A. Torralba, J. Glass. "Unsupervised Learning of Spoken Language with Visual Context". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/82b8a3434904411a9fdc43ca87cee70c-Paper.pdf (cit. on p. 11).

[HWL21] Z. Huang, J. Wu, C. Lv. *Driving Behavior Modeling using Naturalistic Human Driving Data with Inverse Reinforcement Learning*. 2021. arXiv: 2010.03118 [cs.RO] (cit. on p. 7).

[HYC+23] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, H. Li. *Planning-oriented Autonomous Driving*. 2023. arXiv: 2212.10156 [cs.CV] (cit. on pp. 7, 10, 11).

[HZM+22] Y. Hou, X. Zhu, Y. Ma, C. C. Loy, Y. Li. *Point-to-Voxel Knowledge Distillation for LiDAR Semantic Segmentation*. 2022. arXiv: 2206.02099 [cs.CV] (cit. on p. 11).

[KKG+23] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, M. Tancik. *LERF: Language Embedded Radiance Fields*. 2023. arXiv: 2303.09553 [cs.CV] (cit. on p. 18).

[KVB22] M. Klimke, B. Volz, M. Buchholz. "Cooperative Behavior Planning for Automated Driving Using Graph Neural Networks". In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2022. DOI: 10.1109/iv51971.2022.9827230. URL: http://dx.doi.org/10.1109/IV51971.2022.9827230 (cit. on p. 7).

[LLC+21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV] (cit. on p. 20).

[LWL+22] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, J. Dai. *BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers*. 2022. arXiv: 2203.17270 [cs.CV] (cit. on p. 10).

[LWX+22] Z. Li, W. Wang, E. Xie, Z. Yu, A. Anandkumar, J. M. Alvarez, P. Luo, T. Lu. *Panoptic SegFormer: Delving Deeper into Panoptic Segmentation with Transformers*. 2022. arXiv: 2109.03814 [cs.CV] (cit. on p. 10).

[LWZS22] Y. Liu, T. Wang, X. Zhang, J. Sun. *PETR: Position Embedding Transformation for Multi-View 3D Object Detection*. 2022. arXiv: 2203.05625 [cs.CV] (cit. on p. 10).

[LZZ+22] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, K.-W. Chang, J. Gao. *Grounded Language-Image Pre-training*. 2022. arXiv: 2112.03857 [cs.CV] (cit. on p. 12).

[MYVN21] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, F. Nashashibi. "Attention Based Vehicle Trajectory Prediction". In: *IEEE Transactions on Intelligent Vehicles* 6.1 (2021), pp. 175–185. DOI: 10.1109/TIV.2020.2991952 (cit. on p. 17).

[NAZ+22] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, B. Sapp. *Wayformer: Motion Forecasting via Simple & Efficient Attention Networks*. 2022. arXiv: 2207.05844 [cs.CV] (cit. on pp. 7, 9–11, 21).

[NYA+22] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, C. Sun. *Attention Bottlenecks for Multimodal Fusion*. 2022. arXiv: 2107.00135 [cs.CV] (cit. on p. 11).

[PCF+22] L. Peng, Z. Chen, Z. Fu, P. Liang, E. Cheng. *BEVSegFormer: Bird's Eye View Semantic Segmentation From Arbitrary Camera Rigs*. 2022. arXiv: 2203.04050 [cs.CV] (cit. on p. 10).

[RKH+21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV] (cit. on pp. 11, 19).

[SAR18] W. Schwarting, J. Alonso-Mora, D. Rus. "Planning and Decision-Making for Autonomous Vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018), pp. 187–210. DOI: 10.1146/annurev-control-060117-105157. eprint: https://doi.org/10.1146/annurev-control-060117-105157. URL: https://doi.org/10.1146/annurev-control-060117-105157 (cit. on p. 7).

[SCD+19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7 (cit. on p. 37).

[SG05] L. Smith, M. Gasser. "The Development of Embodied Cognition: Six Lessons from Babies". In: *Artificial life* 11 (Dec. 2005), pp. 13–29. DOI: 10.1162/1064546053278973 (cit. on p. 11).

[SJGD22] J. Schmidt, J. Jordan, F. Gritschneder, K. Dietmayer. *CRAT-Pred: Vehicle Trajectory Prediction with Crystal Graph Convolutional Neural Networks and Multi-Head Self-Attention*. 2022. arXiv: 2202.04488 [cs.CV] (cit. on p. 17).

[TL20] M. Tan, Q. V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG] (cit. on p. 21).

[UGA+15] S. Ulbrich, S. Grossjohann, C. Appelt, K. Homeier, J. Rieken, M. Maurer. "Structuring Cooperative Behavior Planning Implementations for Automated Driving". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 2159–2165. DOI: 10.1109/ITSC.2015.349 (cit. on p. 7).

[VSP+23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL] (cit. on pp. 10, 16, 29).

[WJCG22] N. Wu, S. Jastrzębski, K. Cho, K. J. Geras. *Characterizing and overcoming the greedy nature of learning in multi-modal deep neural networks*. 2022. arXiv: 2202.05306 [cs.LG] (cit. on p. 41).

[WLJ+22] X. Wu, Y. Lao, L. Jiang, X. Liu, H. Zhao. *Point Transformer V2: Grouped Vector Attention and Partition-based Pooling*. 2022. arXiv: 2210.05666 [cs.CV] (cit. on p. 11).

[WLL+22] Z. Wang, Y. Lu, Q. Li, X. Tao, Y. Guo, M. Gong, T. Liu. *CRIS: CLIP-Driven Referring Image Segmentation*. 2022. arXiv: 2111.15174 [cs.CV] (cit. on p. 11).

[WQA+23] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, J. Hays. *Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting*. 2023. arXiv: 2301.00493 [cs.CV] (cit. on pp. 8, 23).

[WSDY23] X. Wang, T. Su, F. Da, X. Yang. *ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals*. 2023. arXiv: 2303.12071 [cs.CV] (cit. on pp. 11, 21).

[WTF20] W. Wang, D. Tran, M. Feiszli. *What Makes Training Multi-Modal Classification Networks Hard?* 2020. arXiv: 1905.12681 [cs.CV] (cit. on p. 11).

[YLLS21] J. Yang, S. Lee, W. Lim, M. Sunwoo. "Human-like Decision-Making System for Overtaking Stationary Vehicles Based on Traffic Scene Interpretation". In: *Sensors* 21.20 (2021). ISSN: 1424-8220. DOI: 10.3390/s21206768. URL: https://www.mdpi.com/1424-8220/21/20/6768 (cit. on pp. 9, 29).

[YRLW19] L. Ye, M. Rochan, Z. Liu, Y. Wang. *Cross-Modal Self-Attention Network for Referring Image Segmentation*. 2019. arXiv: 1904.04745 [cs.CV] (cit. on p. 11).

[ZCW+22] T. Zhang, X. Chen, Y. Wang, Y. Wang, H. Zhao. *MUTR3D: A Multi-camera Tracking Framework via 3D-to-2D Queries*. 2022. arXiv: 2205.00613 [cs.CV] (cit. on p. 10).

[ZLC23] J. Zhong, Z. Liu, X. Chen. *Transformer-based models and hardware acceleration analysis in autonomous driving: A survey*. 2023. arXiv: 2304.10891 [cs.LG] (cit. on pp. 10, 11).

[ZSZ+24] X. Zuo, P. Samangouei, Y. Zhou, Y. Di, M. Li. *FMGS: Foundation Model Embedded 3D Gaussian Splatting for Holistic 3D Scene Understanding*. 2024. arXiv: 2401.01970 [cs.CV] (cit. on p. 18).

[ZTG+22] Q. Zhang, M. Tang, R. Geng, F. Chen, R. Xin, L. Wang. *MMFN: Multi-Modal-Fusion-Net for End-to-End Driving*. 2022. arXiv: 2207.00186 [cs.CV] (cit. on p. 10).

[ZWLH23] Z. Zhou, J. Wang, Y.-H. Li, Y.-K. Huang. "Query-Centric Trajectory Prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023 (cit. on pp. 17, 18, 21).

[ZZH+22] H. Zhang, P. Zhang, X. Hu, Y.-C. Chen, L. H. Li, X. Dai, L. Wang, L. Yuan, J.-N. Hwang, J. Gao. *GLIPv2: Unifying Localization and Vision-Language Understanding*. 2022. arXiv: 2206.05836 [cs.CV] (cit. on pp. 11, 12, 19).

Declaration

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

---

place, date, signature