Mathematisch-
Naturwissenschaftliche
Fakultät

Lernbasierte Computer Vision

Masterarbeit

Vision Transformers for Autonomous Driving

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Siddharth Ramrakhiani, siddharth.ramrakhiani@student.uni-tuebingen.de, 2023

Bearbeitungszeitraum: 26.11.2022-26.05.2023

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter: Prof. Dr. Georg Martius, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Siddharth Ramrakhiani (Matrikelnummer 5675683), June 11, 2023

Abstract

Vision Transformers (ViT) have outperformed Convolutional Neural Networks (CNN) in many domains of computer vision, resulting in backbones providing better transfer performance for many tasks. Recently, ViTs have also been employed as vision backbones to obtain excellent reinforcement learning policies for motor control. However, autonomous driving is still an exception where CNNs are the dominant vision backbones. This naturally raises the question – *Can Vision Transformers be good vision backbones and provide better image representations for autonomous driving?* We test this hypothesis by designing experiments to evaluate the performance of Vision Transformers on the closed-loop autonomous driving task in the CARLA simulator. We find that fine-tuning ViTs initialized with MAE weights are crucial for good driving performance, however, still not enough to outperform CNNs. We make all code and data available on `https://github.com/sidr97/vit-autonomous-driving`.

Acknowledgments

First and foremost I would like to thank my parents for always being there and supporting me through thick and thin. I want to thank Kashyap Chitta for not only supervising me throughout this project, but also for discussions that changed the way I approach machine learning problems. I want to thank Prof. Andreas Geiger for providing interesting ideas for this project, and also for the opportunity to work in his group, which has definitely honed my research, presentation and communication skills. I want to thank Prof. Georg Martius for allocating time to be a second supervisor for this project and for interesting discussions during the thesis defense.

Contents

1 Introduction

An autonomous or a Self Driving Vehicle (SDV) is a vehicle that can operate safely without any human intervention. Not only do such vehicles hold promise to transform the way we live and work, but they also offer a lot of benefits. One such benefit is the drastic reduction of traffic related deaths consequentially improving on-road safety – human error makes upto 94% of traffic related deaths every year [1]. Another advantage is a reduction of our carbon footprint, as such vehicles have the potential to reduce harmful emissions by upto 60% [3].

To be able to operate safely and respond to external conditions like a human, the SDV needs to sense its surroundings. This information is fed through various sensors such as cameras, radar sensors, LiDAR sensors, etc. Autonomous driving techniques that use this information to take driving decisions can be broadly classified into 3 categories:

- Modular pipelines
- End-to-end learning
- Direct perception

Modular pipelines decompose the self-driving stack into independent components [49, 33]. This allows for interpretability (as each component can be inspected individually) and parallel development of each as a unit and is therefore used by industry leaders such as Waymo, Uber, Tesla etc.

An alternative approach to this is the **end-to-end learning** method, which directly maps sensory information to driving decisions such as steering, gas, and brake, usually through a neural network [14, 15, 43, 12, 11]. Supervised machine learning and reinforcement learning approaches dominate this paradigm, and the vehicle can directly learn representations to optimize driving behavior. A benefit of this approach is that obtaining data is not as hard, because for example a camera and other sensors can be mounted to a car to collect data automatically. However, such an approach usually doesn't generalize well, and is also less interpretable in comparison to a modular pipeline.

Direct perception is a compromise between the two mentioned approaches. A neural network outputs intermediate representations such as a depth map or a semantic segmentation map, which is then utilized by the vehicle control module to output driving decisions [45, 7, 34]. Although such representations are compact

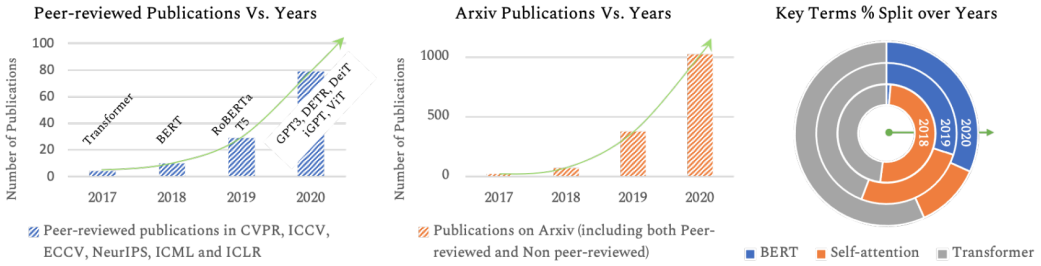

Figure 1.1: There has been a consistent growth of literature regarding transformers in the computer vision community (figure taken from [29])

.

and interpretable, however, jointly training a controller with the neural network is not always easy. Also, as the intermediate representation is a design choice humans need to make, this induces scope of human error into the model.

This work lies in the end-to-end learning paradigm, specifically using **imitation learning** where a set of demonstrations collected from an expert driver are used to optimize driving behavior. Collecting such data in the real world is costly and there exist simulators to circumvent this problem, allowing all training and testing to be done in simulation. This reduces the cost of data collection (the only investment needed is running the software on a computer). We resort to using the open source CARLA simulator in this work [21].

As mentioned above, the SDV needs to sense its surroundings and obtains this information through various sensors. One of the sensors is a vehicle mounted camera, that provides an image of the surrounding scene from a driver's point of view. Until now, all approaches using imitation learning encode this information through a CNN [14, 15, 12, 11, 44, 6]. However, the Vision Transformer (ViT) [20] has shown to achieve superior performance in almost all fields of computer vision [41, 9, 26, 5], leading to better vision backbones. This has sparked a great interest in the computer vision community, with publications involving transformers increasing 8-fold over a period of 4 years (Figure 1.1). However, autonomous driving remains an exception where CNNs are still the dominant vision backbones. This naturally leads to the question – *Can vision transformers be good vision backbones for autonomous driving as well?* We investigate this question in this thesis.

We believe the key reason behind ViTs still not being applied to end-to-end autonomous driving is that ViTs require large-scale data to outperform CNNs [20]. This is generally unavailable in open source autonomous driving datasets. Consequently, ViTs require significantly higher compute power for training. However, He et al. [26] have recently shown a light-weight self-supervised pre-training strategy that allows ViTs to outperform CNNs without requiring large-scale data and also significantly less compute. This strategy has been adopted by [55, 40] to show that simply using

off-the-shelf ImageNet [18] pretrained models proposed in [26] as image encoders and training reinforcement learning policy controllers on top result in state-of-the-art reinforcement learning policies, outperforming supervised training from scratch and also ImageNet initialization. This also suggests that similar training recipes could be applied to end-to-end autonomous driving as well. We therefore investigate if ViTs can be good vision backbones that provide better image representations to optimize driving behavior for end-to-end autonomous driving.

The rest of this thesis is structured as follows. In chapter 2 we list the related work in the fields of end-to-end autonomous driving, transformers in computer vision and pre-training strategies in computer vision. In chapter 3, we describe the background needed to understand the baselines and our proposed approach. Chapter 4 describes the baselines, experimental setup and introduces our proposed approach. Chapter 5 discusses metrics of evaluation and presents the results of our experiments. Chapter 6 concludes the thesis and outlines future research directions stemming from this project.

2 Related Work

In this chapter, we discuss previous literature related to this project. Specifically, we throw light on prior work in end-to-end autonomous driving, transformers in computer vision and pre-training strategies in computer vision.

2.1 End-to-End Autonomous Driving

Initial attempts

Pommerlau et al. [37] pioneered the idea of directly learning control outputs from raw sensory inputs such as images and laser scans, and applied it to follow lanes. Muller et al. [35] followed a similar approach for obstacle detection and avoidance. Bojarski et al. [8] made the first attempt to employ a CNN to map input images from a video sequence to steering commands. Codevilla et al. [16] showed that prior approaches only follow lanes and don't work when a lane change or an ambiguous situation (such as choosing between a left or a right turn at an intersection) occurs, and propose an approach that also accepts control commands to resolve such ambiguities.

Addressing the limitations of initial attempts

Codevilla et al. [17] and Bansal et al. [6] highlight limitations of earlier proposed behavior cloning approaches such as generalization, dataset bias, the causal confusion problem and sample (in)efficiency and propose improvements to address those limitations. [17] specifically highlight dataset bias and high variance. To address these limitations, they modify CIL [16] to include deeper architectures along with a speed prediction head and show that such an architecture performs better. On the other hand, [6] shows that behavior cloning is highly sample inefficient and that such an approach does not result in a robust policy even when 30 million samples are available. They address this limitation by augmenting perturbed samples and creating situations that would otherwise be very sparse in a dataset collected simply by driving.

Recent attempts

Here we discuss the recent attempts, more specifically methods that show evaluation on the CARLA leaderboard as we demonstrate our approach on the same. [12, 10]

tackle the driving problem through a student-teacher approach. [12] decomposes the learning problem in two stages. The first stage learns an agent that has access to privileged information such as the environment layout, position of traffic, etc. In the second stage, this learned agent supervises another agent that only has access to vision information. [10] first learns a model based reinforcement learning policy through offline trajectories (teacher). This policy then provides supervision to a student agent in the form of learned action values (policy distillation). Neural Attention Fields for End-to-End Autonomous Driving (NEAT) [14] is an attention based architecture, posing end-to-end self driving as a dense offset prediction task along with bird's eye view semantic prediction as an auxiliary task. Prakash et al. [38] show that existing sensor fusion methods are not effective in the sense that they do not perform well in complex scenarios, and introduce Multi-Modal Fusion Transformer (TransFuser), also an attention based architecture that integrates complementary sensor (image + LiDAR) representations effectively. InterFuser [46] argues that the style of sensor pair fusion introduced in TransFuser does not scale to multiple sensors and therefore introduces a scalable technique using a transformer encoder-decoder architecture but with enhanced importance to interpretability. All of these approaches operate directly on images i.e., a dense pixel level representation of a scene. PlanT [43] introduces a novel object-level scene representation i.e., using oriented bounding boxes of objects such as vehicles, routes, etc., in the BEV space as input and shows that processing them with a transformer leads to better performance, efficiency and explainability.

All of the above mentioned approaches use CNNs to process the visual input. This work explores the possibility of using vision transformers to process the visual information, potentially providing better image representations for future work. One of the image-only variants of TransFuser, named LatentTransFuser (LTF) is the closest related work to this project, and is one of the baselines considered in our comparisons.

2.2 Transformers in Computer Vision

ViT [20] is the pioneering work that directly employs the transformer [20] architecture on fixed size non-overlapping image patches for image classification, and shows that such architectures perform well when a large amount of training data is available, such as the JFT-300M dataset.

Following this, the computer vision community has seen a cornucopia of vision transformers being applied for various tasks. We list the ones directly related to our work here, and an exhaustive list can be found in [25]. DeiT [48] introduces strategies that allow ViTs to be effective even when high amount of data is not available. [42] adapt transformers for dense prediction tasks and show good performance on monocular depth prediction and semantic segmentation. [31] proposes a hierarchical transformer using a shifted-window approach, and also show that such a model

can potentially serve as a general purpose backbone for computer vision tasks. We use various vision transformer architectures to test how suitable they are as a vision backbone for autonomous driving.

2.3 Pre-Training in Computer Vision

Self-supervised learning (SSL) based pre-training approaches have gained a lot of popularity in computer vision. These approaches can roughly be classified into 2 categories. First are approaches relying on pre-text tasks for pre-training i.e., manually choosing a task for pre-training with a hope that it improves performance on a downstream task [19, 52, 36, 24]. Second are approaches using contrastive learning, that are trained with an objective of being invariant to a set of augmentations chosen manually [27, 50, 22, 56].

A subset of the first category of appraoches use masked image modelling as a pre-training objective. Such approaches aim to reconstruct a given an image with masked pixels [26, 5]. Such techniques have shown to result in excellent reinforcement learning policies for motor control [55, 40]. As autonomous driving can also be posed as a control task, we experiment if such pixel-level pre-training techniques also result in better driving policies.

There also exist alternative approaches that explicitly focus on pre-training for learning abstractions relevant for driving [57, 58, 54]. SelfD [57] is a semi-supervised technique which first uses a small labeled dataset to learn an initial observation-to-BEV policy via imitation, and further uses this learned model to generate pseudo-labels for a larger dataset. Then, another model is trained from scratch on this psuedo-labelled dataset which is further finetuned on the initial labeled dataset. ACO [58] is a self-supervised technique that performs MoCo style pre-training on the image and action space simultaneously. It uses first the Nuscenes driving dataset to learn an observation-to-steering model, uses it to psuedo-label driving scenes from the YouTube driving videos dataset. PPGeo[54] on the other hand is the first pre-training strategy that is completely self-supervised without the need for any pseudo labels. It does so through 2 pre-training stages. The first stage pre-trains a depth and pose estimation module by predicting depth, camera intrinsics and 6-DoF ego motion between 2 given frames. The second stage trains a visual encoder to predict ego motion between past frame and future frame simultaneously, given the current frame. This trains the visual encoder to learn representations relevant for driving, and can then be finetuned on autonomous driving tasks, initializing the visual encoder with these learned weights.

3 Background

This chapter explains the concepts needed to understand baselines, our proposed method and related experiments.

3.1 Autonomous Driving

The term autonomous driving usually refers to the capability of a vehicle to take certain control/maneuvering decisions that would otherwise need a human. These decisions are in the form of lateral (steering) or longitudinal (gas, brake) control. Based on this, autonomous driving has been classified into 6 levels [2] by the Society of Automotive Engineers (SAE):

0. **Driver Only:** The driver needs to completely be aware all the time and must be in control of the car.
1. **Assisted:** The vehicle can exhibit either type of the two mentioned controls, however, the driver must still be on the driving seat and must exercise all types of control.
2. **Partial Automation:** The vehicle can exhibit both types of control in a specific scenario (for example, cruise control on a highway). The driver, although does not have to exercise control, but needs to monitor the system at all times.
3. **Conditional Automation:** The vehicle can exhibit both types of control in a specific scenario and in addition to this, recognizes it's performance limits and requests the driver to resume control within a sufficient time margin. This means that the driver does not need to monitor the system all the time but must be in a position to resume control.
4. **High Automation:** The vehicle can control certain situations completely and the driver is not required in such situations at all. An example of this is the ability to park the vehicle in a parking lot. The driver can simply get out of the car and the car can park itself.
5. **Full Automation:** This is full self-driving and the vehicle can drive itself in all situations. No driver is required at any time.

The approaches for autonomous driving mentioned in the introduction are in the context of full automation. This project also focuses on the full automation category

and deals with the task of point-to-point navigation in an urban setting where the goal is to complete a given route while safely reacting to other dynamic agents and following traffic rules. We use the CARLA simulator 0.9.10 for training and testing.

3.2 The CARLA simulator

This section describes the CARLA simulator to better understand what the data, the input, and output representations looks like.

Developing and testing autonomous vehicles involves a lot of cost and risks. To develop an autonomous vehicle, we first need a vehicle which itself is expensive. On top of that, the vehicle needs to have sensors mounted on it to be able to make sense of its surrounding. Although cameras and radar sensors don't cost as much (costing in the range of a few hundred dollars), the most important sensor in today's AD approaches – the LiDAR sensor, costs thousands of dollars. For example, a G3 Cruise AV is equipped with Velodyne LiDAR sensors, which can even cost upto $64,000 per unit [53]. Secondly, testing an AV is dangerous as mistakes can lead to severe accidents potentially causing loss to human lives and also needing replacement of sensors or the whole vehicle itself, further incurring monetary costs.

An alternative is to develop and test AVs in simulation, before testing and deploying in real facilities or cities. For example, a method showing superior performance in simulation would be a good indicator whether it should be used in a real vehicle or not. Autonomous driving companies like Cruise, Waymo, Wayve, etc., have proprietary in-house simulators for testing their AVs which is not accessible to the general public. We use the publicly available open-source CARLA [21] simulator for this project.

We use the CARLA simulator to be able to make fair comparisons with previously proposed methods, which demonstrate results on the same. CARLA is quite realistic in terms of graphics and simulation of traffic scenarios. It consists of various towns (named Town01 through Town10) and scenarios. An example of town 02 can be seen in Figure 3.1.

CARLA also exposes a powerful API that allows users to control all aspects related to the simulation, including traffic generation, pedestrian behaviors, weathers, sensors and much more. This API can be leveraged to collect data – an expert agent (in terms of CARLA, this means a vehicle that knows how to drive well) can be set up which is then rolled out and made to drive in certain towns/scenarios. This expert agent is equipped with sensors that record the input and save it in memory. An example for this can be a vehicle mounted camera that records images from the driver's point of view. Therefore, collected data essentially are sensor readings at various timesteps along with driving decisions made at those timesteps.

CARLA also gives access to privileged information, which means expert decisions

Figure 3.1: Town 02 of CARLA

at the recorded timesteps, in the form of driving decisions (steering, gas, brake) or **waypoints**. Waypoints are the position of the ego-vehicle at future timesteps, represented as (x, y) coordinates in the BEV space. For example, waypoints for 4 time steps would be represented as $w_t = \{(x_i, y_i)\}_{t=0}^{3}$. Figure 3.2 gives a visualization of how this would look like. The red box denotes the ego-vehicle with the blue dots representing waypoints. The first waypoint denotes the position of the ego-vehicle at $t = 0$ meaning the current position of the ego-vehicle, which is always the origin (0,0) as this BEV space is in the coordinate frame of the ego-vehicle. This thesis uses waypoints as the output representation, which is further passed on to a low-level PID controller that outputs driving decisions.

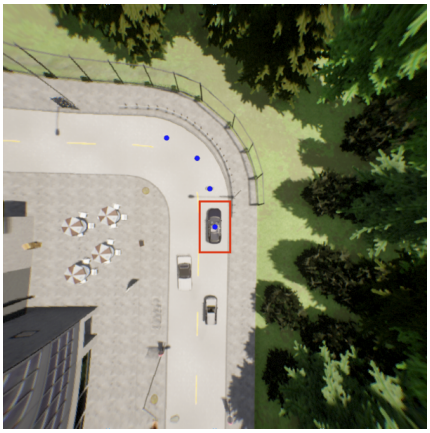

Figure 3.2: Waypoints in CARLA

3.3 Imitation Learning for Autonomous Driving

Given a set of expert demonstrations, the goal of imitation learning (IL) is to learn a policy that mimics this behavior. We define a policy as a function/mapping from inputs to waypoints. These waypoints are then provided to a low-level controller

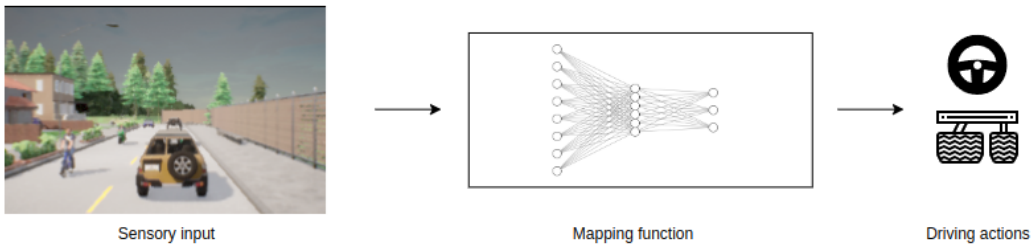

Figure 3.3: Imitation Learning directly maps sensory input to driving decisions

such as a PID controller that then outputs actions accordingly. The set of expert demonstrations can be represented as a dataset $\mathcal{D} = (\mathcal{X}_i, \mathcal{W}_i)_{i=1}^{Z}$ of size Z. $\mathcal{X}$ contains sensor information that can be interepreted as high dimensional observations of the environment. In our case, the sensor information includes 3 RGB cameras, one facing forward, and the others at an angle of 60° towards the left and right. $\mathcal{W}$ contains the expert actions denoted as $\mathcal{W} = \{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^{T}$ in the form of 2D waypoints in the Birds-Eye-View (BEV) space. The policy π is trained in a supervised fashion using a loss function $\mathcal{L}$:

$$argmin_{\pi} \mathbb{E}_{(\mathcal{X},\mathcal{W})\sim\mathcal{D}}[\mathcal{L}(\mathcal{W}, \pi(\mathcal{X}))]$$

The policy π is usually a neural network designed in a manner to input the 3 RGB camera information and output waypoints (Figure 3.3). This mapping function involves human design choices. For example, transfuser [15] involves a lot of auxiliary tasks such as depth map estimation, semantic segmentation, BEV bounding box detection, HD map generation, etc. Another example is NEAT [14] which does 2D BEV semantic prediction as an auxiliary task. A part of the mapping function involves processing RGB camera images. Current literature and state-of-the-art techniques do this using a Convolutional Neural Network (CNN), for example Resnet-34 [28] in the case of NEAT. However, vision transformers are used as an image encoder in many fields of computer vision and have shown to outperform CNN based backbones. Recently, vision transformers have been shown to provide better image representations even for control tasks [55, 40]. Our literature review reveals that autonomous driving is still an exception and therefore this forms the motivation of this thesis. The rest of this chapter describes the transformer and vision transformer architecture.

3.4 The Transformer architecture

The architecture of the transformer model [51] is shown in Figure 3.4. It was first demonstrated on the machine translation task in natural language processing. It accepts a sequence of inputs $(x_1, x_2 \ldots x_n)$ (A sentence in English for example) and

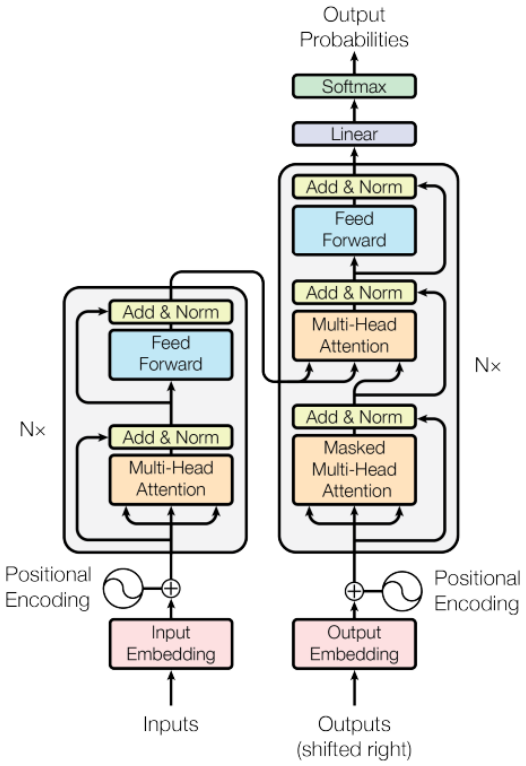

Figure 3.4: The transformer architecture

processes them through an encoder-decoder style architecture to output a sequence $(y_1, y_2, \ldots y_n)$ (A sentence translated to French). For this thesis, we are only concerned with the encoder and therefore only explain components until the encoder. A full explanation of the transformer can be found in [4]. There are 3 key components architecturally that contribute to the success of transformers:

- Self-Attention
- Multi-Head Attention (multiple self-attention layers)
- Positional Encodings

3.4.1 Self-Attention

As explained above, the transformer architecture accepts an input sequence $(x_1, x_2 \ldots x_n)$. These inputs are first linearly projected to a dimension d and stacked together, resulting in a matrix $X \in \mathbb{R}^{n \times d}$. Attention, as described in [51] is mapping a query and a set of key-value pairs to an output, where the query, key, value and output are all vectors. Self-attention can then be described as attention over all items of the

matrix X i.e., an output associated to each item $X_i \forall i \in \{1 \dots n\}$. The calculation of these outputs is described below:

- Define 3 learnable weight matrices $W^Q \in \mathbb{R}^{d \times d_k}$, $W^K \in \mathbb{R}^{d \times d_k}$, and $W^V \in \mathbb{R}^{d \times d_v}$.
- Multiply X with W^Q, W^K and W_V to obtain query Q, key K and value V.
- Perform attention calculation according to the following equation:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3.1}$$

The essence behind calling this method "self-attention" is that when attention is calculated for a specific element, all other elements of the input sequence are considered. This also means that the output will contain information about the global context of the input and therefore will not have problems in capturing long range context, a problem that traditional sequence models suffer from [59].

3.4.2 Multi-Head Self Attention

Instead of being restricted to just one self-attention layer, multi-head self attention (MHSA) means employing multiple self-attention layers or "heads" and fusing their outputs together. This allows the model to realize various types of relationships that might exist in the sequence, which potentially just one self-attention head would not be able to capture. The process of calculating MHSA is described below:

- Initialise h self-attention layers, meaning initialising matrices $\{W_i^Q, W_i^K W_i^V\} \forall i \in \{1 \dots h\}$
- For each self-attention layer, calculate $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$
- Initialise $W^O \in \mathbb{R}^{hd_v \times d}$.
- $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$

The matrix W^O is introduced in order to keep the dimensionality of $\text{MultiHead}(Q, K, V)$ the same as $\text{Attention}(Q, K, V)$.

3.4.3 Positional Encodings

Traditional sequence-to-sequence models take into account location information as they process the input one after the other. For example, in recurrent neural networks (RNNs), the hidden state at one time-step takes into account the output of the hidden-state at the previous time-step. However, this is not the case with transformers and every operation introduced above can be carried out even if the order of elements in the sequence changes. Therefore, the model treats the input not

as a sequence but rather a set. To be able to make sense of absolute positions, this information needs to be injected into the model to represent location information.

To do this, one could simply assign integers $(0,1,2,\ldots)$ representing the index of the location. However, one big problem associated with this approach is that the model can encounter sequences longer than the ones present in the training data. To overcome these problems, authors in [51] propose to use sine and cosine functions of different frequencies resulting in vector positional encodings that are unique for each time-step:

$$PE_{(pos,2i} = sin(pos/10000^{2i/d})$$
$$PE_{(pos,2i+1} = cos(pos/10000^{2i/d})$$

where pos is the position and i is the index of the location.

Another way of incorporating location information can be to use learned embeddings, however the authors found the results to be nearly the same with learned embeddings and sinusoidal embeddings and chose the latter as it could allow the model to generalize to sentence lengths not encountered in the training set.

3.5 Vision Transformers

The first vision transformer was introduced by Dosovitskiy et al. [20], which achieved state-of-the-art performance on ImageNet classification. Since then, lot of work has been done to employ such architectures for various tasks in computer vision such as object detection [9], depth estimation and semantic segmentation [41]. A lot of research has also gone into self-supervised pre-training of vision transformers which reveals some (simple) particular recipes work exceptionally well. We describe all of it below.

3.5.1 The ViT architecture

The architecture of the first ever vision transformer – ViT, is illustrated in Figure 3.5. There transformer encoder is a vanilla transformer introduced in [20] and described above. Apart from this, there are 3 key ideas to understand:

- Treating images as a sequence
- Extra learnable [class] embedding
- MLP head

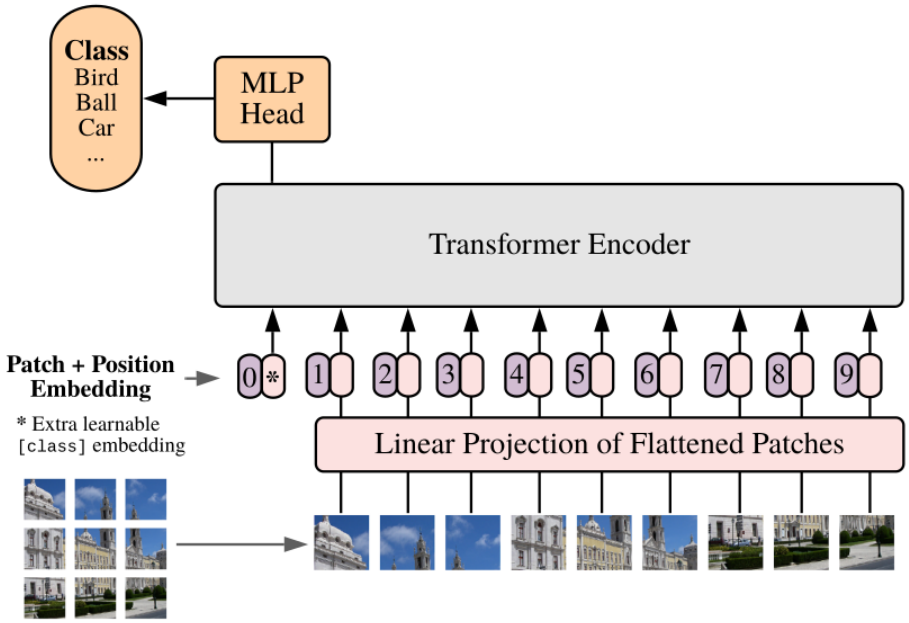

Figure 3.5: The Vision Transformer [20] architecture

3.5.2 Treating images as a sequence

As mentioned in Section 3.4, a transformer accepts a sequence $(x_1, x_2, \ldots x_n)$ as input. To employ such an architecture for computer vision, an image is split in patches (authors proposed patches of 16×16 pixels), and treated as a sequence. The standard transformer receives a 1D input for processing. To handle 2D patches, the authors reshape the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches. If the patch size is $p \times p$, this would result in $N = HW/P^2$ number of patches. Therefore the sequence of patches can be represented as $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2.C)}$. These patches are unrolled to a 1D vector, and further projected to a constant latent dimension D through a linear layer. The output at this stage is termed as patch embeddings.

3.5.3 Extra learnable [class] embedding

Once patch embeddings are obtained, an extra embedding is added, particularly for using its output later. One can call this an arbitrary embedding, but because its output is specifically used later for classification, it is called the [class] embedding. Consequentially, there are $(N+1)$ embeddings now that are added with positional embeddings (explained above in section 3.4.3) and input to a vanilla transformer encoder.

3.5.4 MLP head

As mentioned above, $(N+1)$ patch embeddings are processed through a transformer encoder to result in $(N+1)$ output embeddings. A linear MLP layer then takes the output of the [class] embedding token and uses it to obtain class probabilities. The authors also try a different variant – average pool the output of the original N embeddings and report that both variants perform equally well. The only caveat is that the average pooling method requires a different learning rate.

3.6 Swin Transformer

ViT [20] is an attempt to apply the vanilla transformer [51] directly for image classification with as few modifications as possible. The promising performance of ViT indicates that transformer architectures can be used for other computer vision tasks as well. However, ViT only deals with image classification and there exist challenges when adapting such a model for other dense vision tasks such as object detection and semantic segmentation. For example, the quadratic complexity of self-attention makes operating on higher resolution images intractable. Therefore, Liu et al. [31] propose a general purpose transformer backbone for computer vision that overcome these challenges. The architecture of Swin Transformer is illustrated in Figure 3.6. There are 2 key components to understand Swin Transformer:

- Swin Transformer Block
- Patch Merging

We first give a brief overview of the architecture, followed by a description of the two mentioned components for a better understanding.

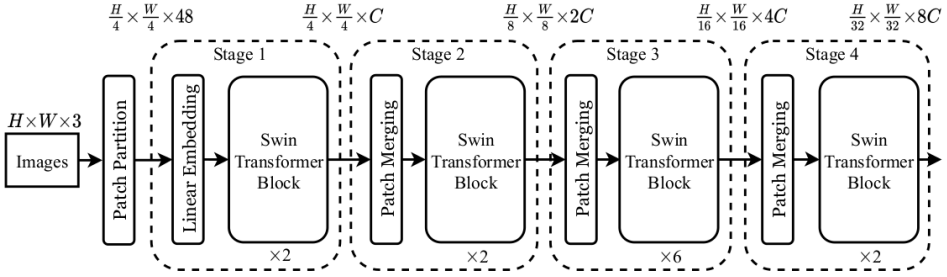

Figure 3.6: The architecture of the Swin Transformer

3.6.1 Overview

The input to Swin Transformer is an image, which is split into non-overlapping patches and each patch is treated as a token, similar to ViT. However, there are two

key differences here in comparison to ViT:

- Attention in Swin Transformer is computed locally within patches, similar to a CNN. This is a stark difference in comparison to ViT, which performs global self-attention over the input patches.
- The patch resolution in Swin Transformer is much lower – 4×4 pixels as opposed to 16×16 pixels in ViT.

Several modified transformer blocks called Swin Transformer blocks then process these patches, with patch merging happening after every block.

3.6.2 Swin Transformer Block

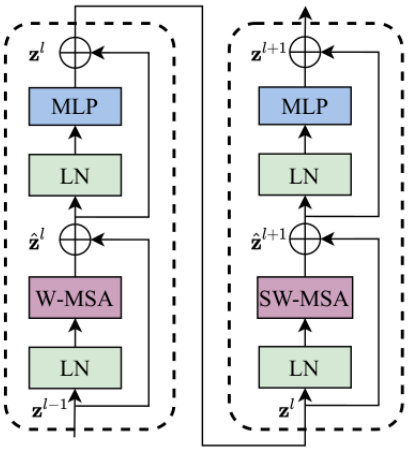

Figure 3.7: Two successive Swin Transformer blocks

Different from the vision transformer encoder, attention in a swin transformer block is computed as follows:

$$\hat{\mathbf{z}}^l = \text{W-MSA}(\text{LN}(\mathbf{z}^{l-1})) + \mathbf{z}^{l-1},$$
$$\mathbf{z}^l = \text{MLP}(\text{LN}(\hat{\mathbf{z}}^l)) + \hat{\mathbf{z}}^l,$$
$$\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}(\text{LN}(\mathbf{z}^l)) + \mathbf{z}^l,$$
$$\mathbf{z}^{l+1} = \text{MLP}(\text{LN}(\hat{\mathbf{z}}^{l+1})) + \hat{\mathbf{z}}^{l+1},$$

Here, MSA means Multi-Head Self Attention which essentially is the regular manner in which self-attention is computed, again similar to ViT. SW-MSA stands for Shifted-Window Multi-Head Self Attention, which computes self-attention on a shifted window configuration. MSA and SW-MSA can be visualized in Figure 3.8. The first part of the swin transformer block (layer l) computes attention on a regular configuration. The second successive block then applies attention on a a shifted

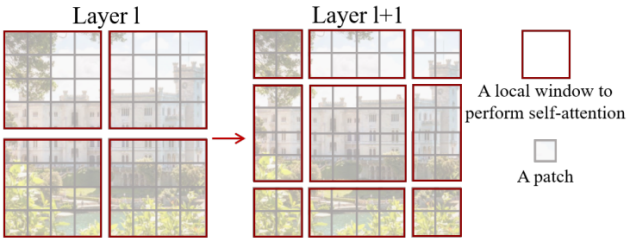

Figure 3.8: Layer$l$ denotes regular configuration for computing self-attention (W-MSA), and layer $l+1$ denotes the shifted window configuration (SW-MSA)

patch configuration (layer $l+1$). This can be seen as a CNN-like inductive bias being introduced in the model by allowing cross-window connections with the previous layer and enhances the modelling power of Swin Transformers.

3.6.3 Patch Merging

In order to produce hierarchical feature representations, the number of tokens are reduced after every Swin Transformer block by simply grouping 2×2 neighboring patches. This results in a single patch but with $4C$ features (if C was the number of features before merging). A linear layer is then applied to reduce the feature dimensions from $4C$ to $2C$. Such a hierarchichal structure is also similar to a CNN, where the receptive field decreases and feature depth increases as the network gets deeper.

3.7 Pre-training for Vision Transformers

The state-of-the-art performance of the ViT described above is achieved by pre-training of ViT-L on the JFT-300M [47] dataset followed by supervised training on the ImageNet dataset. In fact, the model accuracy drops by 13% on the ImageNet test set if directly trained on ImageNet without any pre-training. The authors highlight that ViTs outperform CNNs only when large scale training data is available (for example, the JFT-300M dataset has about 375M images). This is because CNNs encode prior knowledge through inductive bias (for example, translational equivariance) whereas transformers have no such inductive bias and need to learn this during training, thereby requiring significantly more samples than CNNs. We have experimented with self-supervised pre-training strategies introduced in [26] and describe them below.

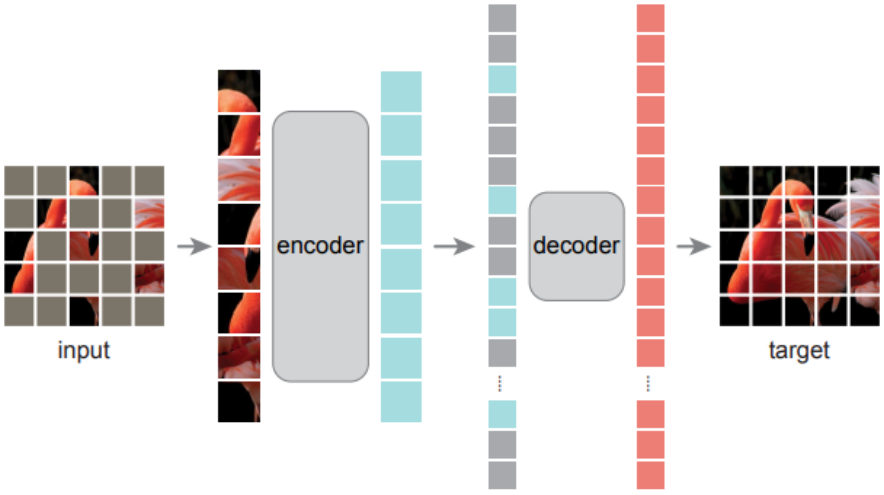

Figure 3.9: The MAE architecture introduced by He et al. [26]

3.7.1 Masked Autoencoders are Scalable Vision Learners (MAE)

He et al. [26] proposed a conceptually simple yet efficient pre-training strategy for learning high capacity models that generalize well. Figure 3.9 illustrates this idea. An image is given as input to an encoder-decoder style architecture with some patches masked and the task is to reconstruct the missing patches. The encoder is a ViT and the decoder is a lightweight transformer that just needs to reconstruct the original image from the latent representation of the encoder. The input to the encoder are the full set of visible (non-masked) patches. The input to the decoder are all the patches (restored back to their original positions). The authors find that masking a high proportion of the image, for example 75% (i.e., only $1/4^{th}$ of the patches are visible) results in optimal performance. This proves to be an excellent self-supervised pre-training strategy allowing the model to learn general visual representations as:

- As mentioned above, the encoder only processes the visible patches. A high masking ratio results in the encoder processing a minority of input patches making this technique scalable and efficient (The authors report a 3× speedup in pre-training time).
- There is no need of dataset specific augmentations, as random sampling of masks itself serves as a dataset-independent data augmentation strategy.
- There is no assumption of any sorts on the distribution of the training set, allowing the pre-training phase to have as many and as diverse images.

The authors show that such a strategy alleviates the need for training ViTs on JFT-300M and self-supervised pre-training on the IN-1K dataset followed by supervised fine-tuning again on the same dataset results in state-of-the-art performance. They hypothesize that the reason behind this strategy performing so well is that masking

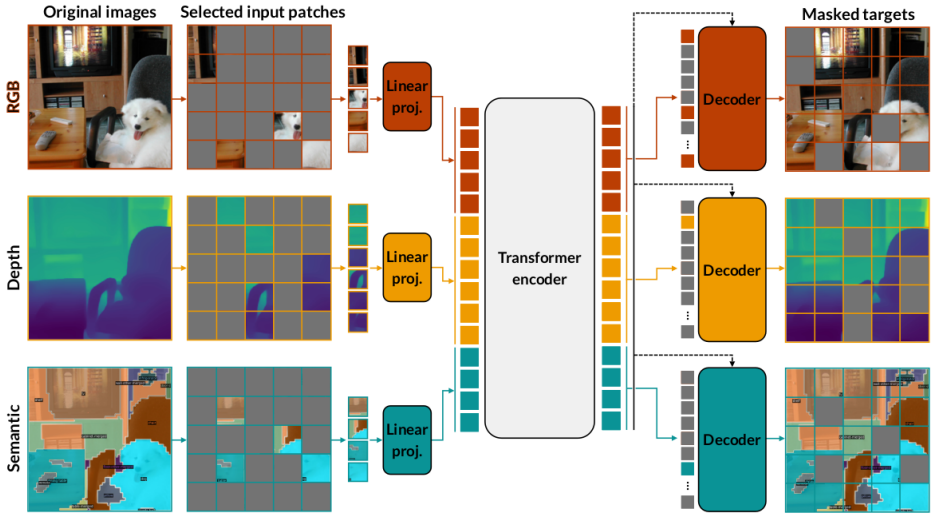

Figure 3.10: The MultiMAE architecture

a high number of patches reduces redundancy and that reconstruction becomes a harder tasks that requires the model to capture representations beyond a low-level understanding of the image.

3.7.2 MultiMAE: Multi-modal Multi-task Masked Autoencoders

Bachmann et al. [5] argue that (1) MAE is restricted to only RGB modality and that in practice there are often multiple modalities available (for example, depth) that can be leveraged and (2) pre-training with diverse tasks has shown to improve performance on downstream tasks . They therefore propose a pre-training strategy that extends MAE to a multi-modal multi-task setup. Instead of simply reconstructing/inpainting an RGB image given a masked

This setup is multi-modal as it operates on multiple modalities apart from RGB (they use depth and semantics in their experiments) and multi-task as the task is to reconstruct all modalities from the given sparse input patches. The authors show that this pre-training strategy outperforms MAE on downstream tasks. They also show that in the case when only RGB modality is available, usage of pseudo-labeled depth and semantics also outperform MAE, making this strategy widely applicable on any RGB dataset. The pre-training strategy and architecture of MultiMAE is shown in Figure 3.10.

3.7.3 Encoder

The inputs to MultiMAE are partially masked RGB, depth and semantic (ground-truth or psuedo-labeled) images. Each image is then split into non-overlapping

patches and only the visible patches (non-masked) are projected to a latent dimension using a linear projection separately for each modality. The tokens resulting from the linear projection layers are then concatenated and input to a transformer encoder.

3.7.4 Decoder

The task of the decoders is to reconstruct each modality from the latent representation of the encoder. The input to each decoder are (1) output tokens from the encoder of the respective modality and (2) masked out patches from the initial input, reshuffled to their original positions just as in MAE. To incorporate information from tokens of other modalities, a single cross-attention layer is added in each decoder using modality output tokens as queries and all output tokens as keys/values. As the decoder is also a transformer, positional embeddings are added to the query tokens before being input to the decoder. Just as in MAE, the decoders are lightweight and the encoder is the compute-heavy part of the model.

4 Method

In this chapter, we explain the problem setting, baselines, and finally our proposed method to test if vision transformers can be good image encoders for autonomous driving.

4.1 Problem Setting

The task at hand is to develop an autonomous driving agent that can drive through a set of predefined routes in the realistic 3D simulator CARLA, version 0.9.10.1 [21]. For each route, agents are initialized at a starting point and directed to drive to a destination point, provided with a description of the route through GPS style coordinates, map coordinates or route instructions. Routes are defined in a variety of situations, including freeways, urban areas, residential districts and rural settings. While driving, agents face multiple traffic scenarios for example lane changing, lane merging, handling traffic lights and signs, negotiations at traffic intersections and roundabouts, etc.

In the context of learning based end-to-end autonomous driving pipelines, image information needs to be encoded to obtain a relevant low-dimensional feature embedding that can be used for further processing. All methods until now use CNNs to encode image information [46, 14, 12, 15, 6]. However, ViTs have also shown to perform better than, or atleast on par as CNNs on various computer vision tasks [41, 20, 9] and consequently result in vision transformer based image encoding backbones that can be used for better transfer learning on other tasks or datasets. This naturally leads to the question – *can vision transformers prove to be good vision backbones for end-to-end autonomous driving as well?* We explain our proposed approach to investigate this question in this section.

Figure 4.1 shows the experimental setup for the same. CARLA provides RGB images from 3 cameras – one facing the front, and two oriented at an angle of 60° to the left and right respectively. We design an encoder-decoder style driving architecture that accepts the 3 RGB images as input and outputs driving decisions. Choosing an architecture-agnostic vision backbone allows us to not only use ViTs but also compare ViTs directly with vision backbones used in state-of-the-art autonomous driving pipelines (for example, RegNetY in [15]). The decoder is a simple, lightweight autoregressive GRU waypoint decoder. This results in the vision backbone majorly

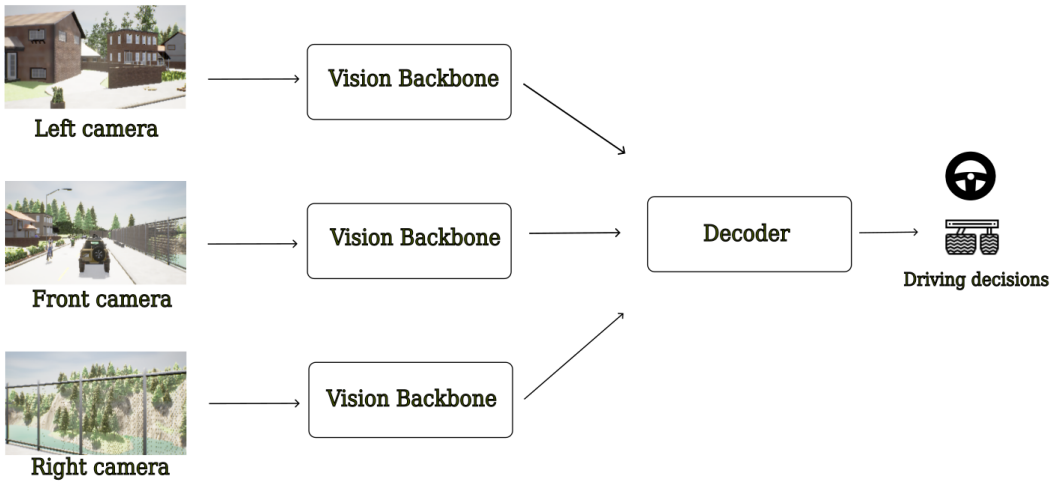

Figure 4.1: Experimental setup

contributing to the expressiveness of the model and therefore all results can be attributed to the choice of the backbone.

4.2 Baselines

We compare our method with two baselines (1) Latent TransFuser – an image only version of TransFuser introduced in [15] and (2) AIM-CNN – a modified version of AIM (introduced in [38]).

4.2.1 Latent TransFuser

TransFuser is an end-to-end driving method proposed by Chitta et al. [15], which had outperformed all prior methods on the CARLA leaderboard. It takes two modalities as input – an RGB image (from the left, front and right cameras) and a LiDAR Bird's Eye View scan. The two modalities are processes through modality specific branches, namely Image branch and BEV branch. However, the method also enforces interaction of the two modalities via transformers at various scales. The final feature vectors of the two modalities are then concatentated together to obtain one final 512-dimensional feature vector. This feature vector then serves as an input to an auto-regressive GRU waypoint decoder, which outputs waypoints.

This network is trained using an L_1 loss objective between ground truth and predicted waypoints. Besides predicting waypoints correctly, motivated by [12], TransFuser also has auxiliary losses through 4 auxiliary tasks (figure 4.3):

- 2D depth and semantics: A depth estimation and semantic segmentation of

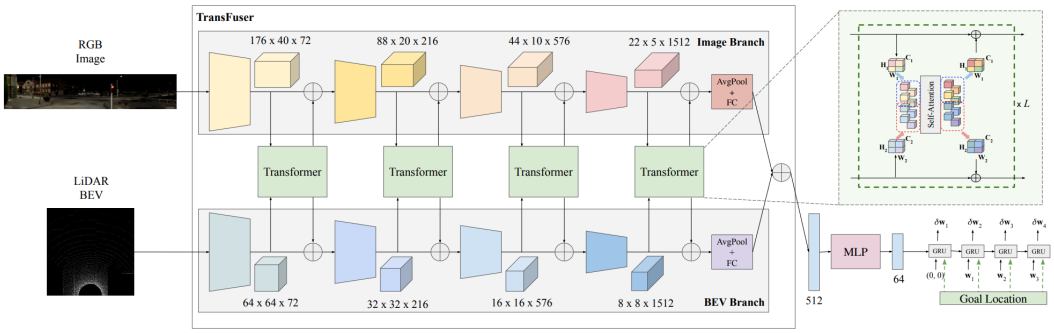

Figure 4.2: The architecture of the TransFuser method

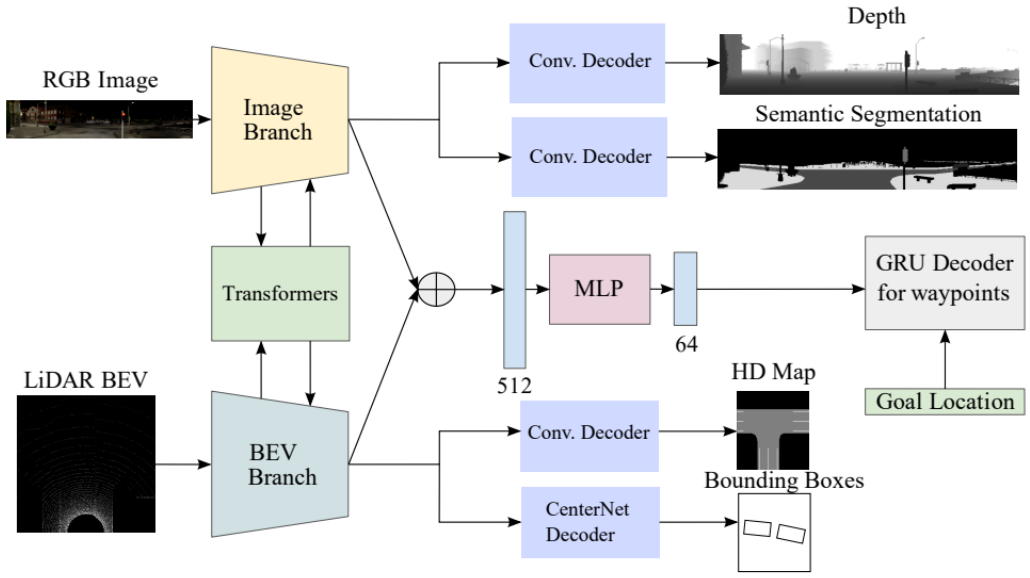

Figure 4.3: TransFuser uses 4 auxiliary tasks along with waypoint estimation

the front-view RGB images.

- HD map: Predicting a 3 channel BEV segmentation mask containing road, lane marking and other classes.
- Bounding boxes: Locating other vehicles in the scene from a BEV perspective.

Latent TransFuser is an **image only** version of TransFuser, which replaces the 2 channel LiDAR BEV input with a 2 channel positional-encoding of identical dimensions. The 2D positional encoding is a grid consisting of values $[-1, 1]$ on the right-left and top-down axis, split into 256 intervals to result in a 2 channel image of resolution 256×256. Apart from this change, the training procedure, loss functions and auxiliary tasks all remain identical to the TransFuser setup.

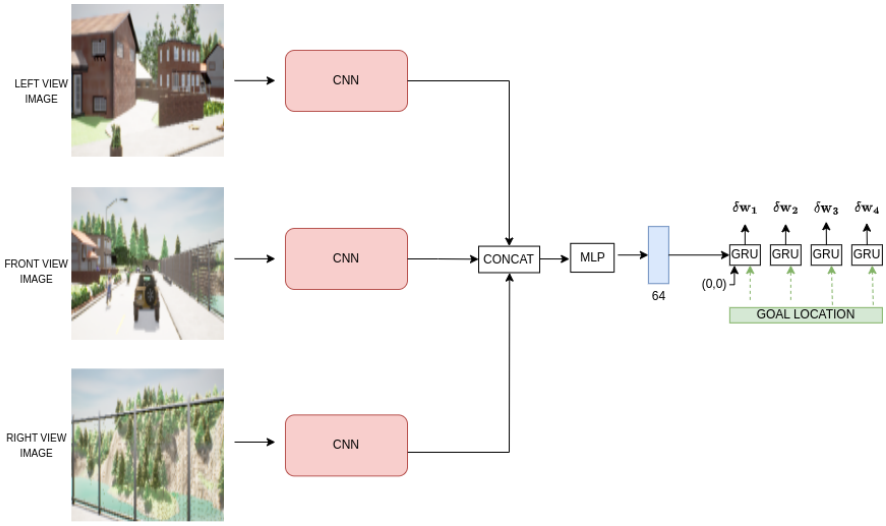

Figure 4.4: AIM-CNN: The modified AIM baseline

As our experimental setup does not use auxiliary tasks, we re-train Latent TransFuser without auxiliary tasks and the only loss being an L_1 loss between predicted and ground-truth waypoints to serve as a baseline.

4.2.2 AIM-CNN

Auto-regressive IMage-based waypoint prediction (AIM) was a baseline used in [38], which takes in a front camera image, processes it through a ResNet-34 encoder and passes the output to an auto regressive GRU decoder to predict waypoints. We modify AIM by using 3 camera images (left, front, right) as input, processing each image through a view-specific CNN, concatenating the output to serve as an input to an auto regressive GRU waypoint decoder. Figure 4.4 illustrates this. We call this baseline AIM-CNN.

4.3 AIM-ViT: Driving with Vision Transformers

4.3.1 Proposed architecture

As the scope of this thesis is to study the effectiveness of vision transformers as an image encoder, we design a driving agent that exclusively tests the power of vision transformers as vision backbones without the interference of any other factor. Figure 4.5 illustrates this architecture.

The input to the agent are three RGB images – one from the camera facing the front, and two from cameras oriented at an angle of 60° to the left and right respectively. These are then split in patches and passed through camera specific

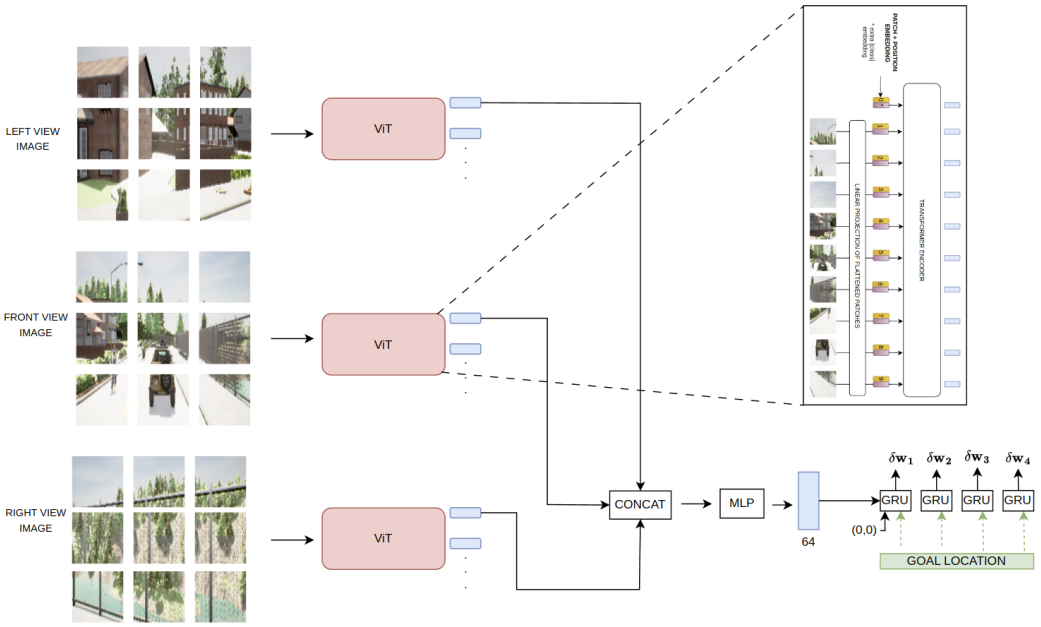

Figure 4.5: Proposed architecture

vision transformers. The output features of the [class] token from the three views are then concatenated and passed through an MLP layer resulting in a 64 dimensional feature vector. This feature vector serves as an input to the GRU waypoint decoder, which we have directly borrowed from [15]. This is trained in a supervised fashion, using an L_1 loss between predicted and ground-truth waypoints. We experiment with different types of transformer encoders and list a description of each below:

- AIM-ViT: Vision Transformer encoder proposed in [20] (ViT-Tiny, ViT-Small, ViT-Base).
- AIM-Swin: Swin Transformer encoder proposed in [31] (Swin-Tiny, Swin-Small, Swin-Base).
- AIM-ViT-MAE: Vision Transformer encoder proposed in [20], pre-trained with the strategy introduced in [26].
- AIM-ViT-MultiMAE-IN1K: Vision Transformer encoder proposed in [20], pre-trained on ImageNet-1K dataset with the strategy introduced in [5].
- AIM-ViT-MultiMAE-Perception: Vision Transformer encoder proposed in [20], pre-trained using perception training strategy (described below).

We train all ViT and Swin architectures using the same training method and recipes as in [5]. We notice that these strategies are highly important to stabilize training of vision transformers and getting them to converge.

4.3.2 Perception Training: Additionally pre-training MultiMAE on CARLA

As mentioned above, the encoder in AIM-ViT-MultiMAE-IN1K is pre-trained using the strategy introduced in [5] on the ImageNet (IN-1K) dataset. This includes pre-training on RGB, depth and semantic segmentation (semseg) modalities. However, when tested on images that the agent would witness during driving in CARLA, the reconstructions are very poor indicating that the encoder cannot generalize to the CARLA domain. An example of this is shown in Figure 4.6, where the input is only the RGB modality, as would be the case with the AIM-ViT agent during training and evaluation.

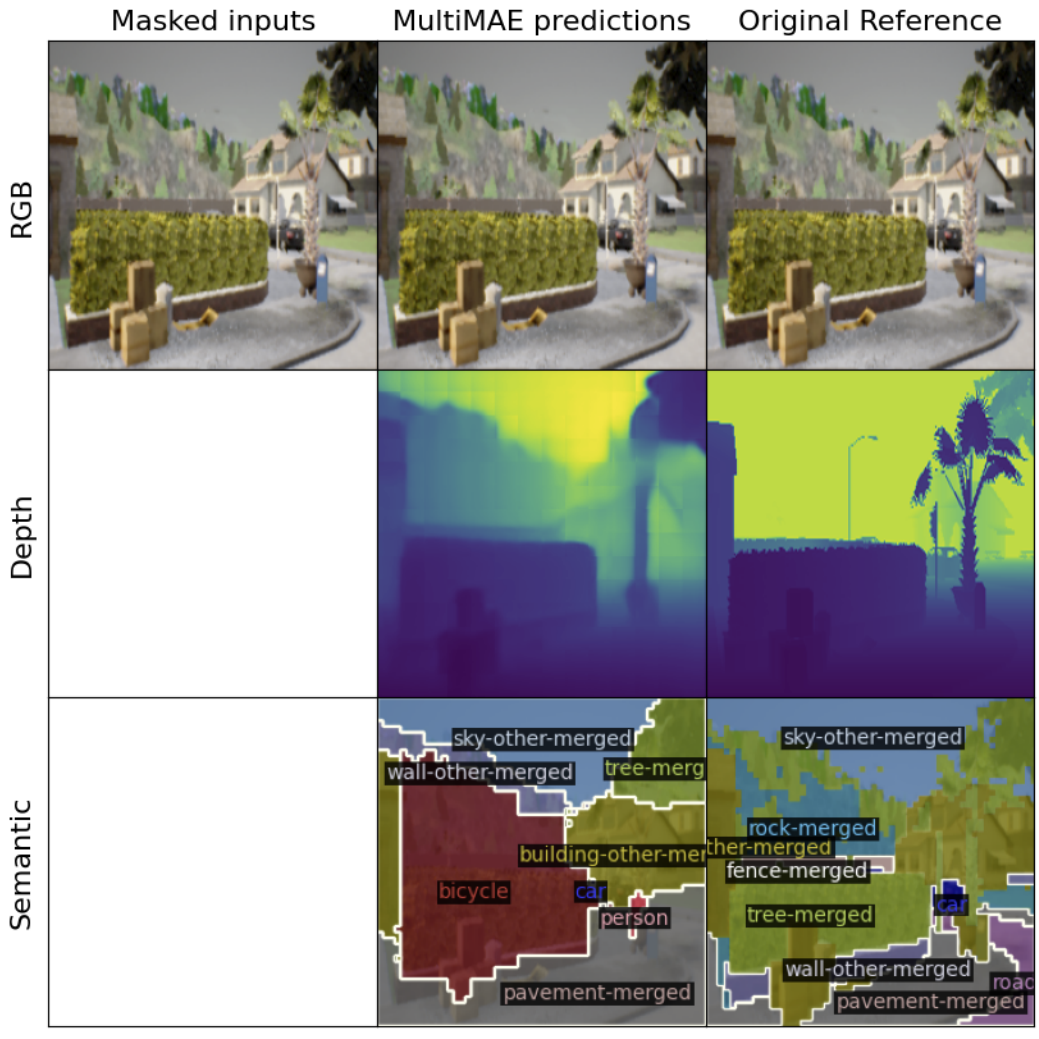

Figure 4.6: Predictions of MultiMAE on CARLA (pre-trained on IN-1K).

It has been shown that source domain pre-training data has a big impact on the performance of such encoders [55, 40]. To help the encoder adapt to the CARLA domain, we further pre-train the encoder on our training dataset, and call this phase perception training. Although not a strict indicator of driving performance, we see an improvement in the reconstructions of this addtionally pre-trained encoder suggesting that it has adapted to the CARLA domain. This can be seen in Figure 4.7, where the input is the same as in Figure 4.6, but the predictions have improved drastically.

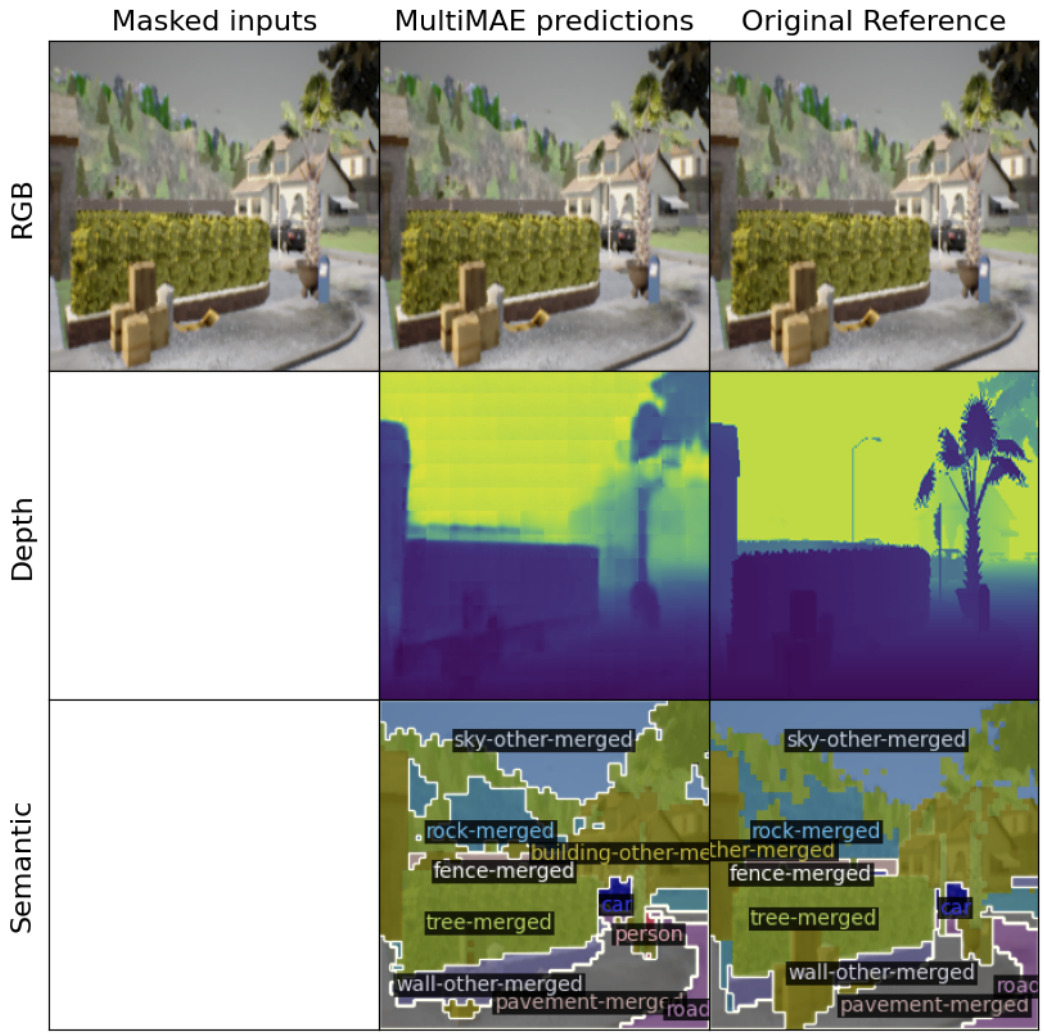

Figure 4.7: Predictions of MultiMAE further pre-trained using perception training strategy.

Details on mapping semantic segmentation classes

The weights of MultiMAE provided by the authors [5] is pre-trained on IN-1K dataset using RGB, depth and semseg modalities. As IN-1K only has RGB images, the authors obtain depth and semseg modalities via psuedo-labeling. For depth, they use a DPT-Hybrid [41] that was trained on Omnidata [23]. For semseg, they use Mask2Former [13] model with a Swin-S backbone [31] trained on COCO [30].

As we are further pre-training MultiMAE on our CARLA data, we need to make sure that the depth processing and class indexing of semantic classes is kept consistent with IN-1K. Depth images need no special treatment as they can be standardized the same way it has been done in [5]. However, the indexing of semantic segmentation classes of CARLA is not the same as the indexing of the pseudo-labeled classes of IN-1K (i.e., semantic classes occuring in COCO) that the MultiMAE encoder was pre-trained on. For example, the class "Pedestrian" has index 4 in CARLA, whereas it is 0 in COCO. Moreover, there exist classes in CARLA that don't exist in COCO, for example, CARLA has the class "Green light" which does not exist in COCO (there exists just 1 class named "Traffic light"). For this, we pseudo-label a small subset of our dataset, using the same pseudo-labeling setup as [5]. Once we have the pseudo-labels, we compare these with ground-truth CARLA semantic segmentation labels, and determine a mapping through majority voting over pixels in the whole dataset. For example, if a class x_{carla} is mapped to a class x_{coco}, then pixels belonging to the class x_{carla} in the subset have the maximum pixels belonging to the class x_{coco}.

5 Experiments

This section consists of a description of training and evaluation settings, metrics for evaluation, obtained results, and comparisons with baselines.

5.1 Training data

Our training dataset is borrowed from TransFuser (Chitta et al. [15]) which contains 168k frames from Town 01 through Town 10 (except Town 02 and Town 05, which have 59k frames and are used for evaluation). This way, we have a dataset that can be compared to prior state-of-the-art techniques. Moreover, as TransFuser is trained on this data, it also gives a guarantee that the dataset size is enough for an agent to learn driving in CARLA.

5.2 Evaluation

We evaluate our proposed method and baselines on the CARLA [21] simulator, on open-loop and closed-loop metrics (described below). We use the challenging LAV benchmark [11] for evaluation. It consists of testing on 4 routes in Town 02 and Town 05, in 4 different weather scenarios (ClearNoon, CloudySunset, SoftRainDawn HardRainNight) resulting in a total of 16 routes. For our experiments, we report results over 1 evaluation run as the score differences between competing models is high. For greater confidence on our most important results, we report the mean and standard deviation over 3 evaluation runs.

5.2.1 Open-loop metrics

The term open-loop refers to offline evaluation, i.e., using a dataset to evaluate decisions of an agent w.r.t the ground truth. One example of this (which we also use in our experiments) is the absolute error between predicted and ground-truth waypoints. A description of the same, with relevance to our experiments is given below:

- **Absolute error (L1) between waypoints**: For every frame in the validation/testing dataset, images from the left, front and right cameras are given as input, and

the agent predicts waypoints. This is compared to the ground truth waypoints using the following equation:

$$V = \frac{1}{N} \sum_{i=1}^{N} |(w_t)_i - (\hat{w_t})_i|$$

5.2.2 Closed-loop metrics

The term closed-loop refers to online evaluation i.e., the agent is made to drive in the simulator on various towns (Town 02 and Town 05 in our case) with various traffic scenarios. The goal here is to complete the route in as few infractions as possible. We use the Driving Score **(DS)**, Infraction Score **(IS)** and Route Completion **(RC)** for evaluating the proficiency of agents in closed-loop driving. A more detailed explanation of the metrics is as follows:

- **Route Completion**: Percentage of the route distance completed by an agent. If R_i is the percentage of route distance completed by the agent in route i, and there are N routes in total, Route Completion would be calculated as:

$$\text{RC} = \frac{1}{N} \sum_{i=1}^{N} R_i$$

- **Infraction Score**: Every agent start with a score of 1.0. For every infraction occurring, there is a penalty that gets multiplied. A few penalty scores are:
  - Collisions with pedestrians – 0.50
  - Collisions with other vehicles – 0.60
  - Running a red light – 0.70
  - Running a stop sign – 0.80

  A full list of penalty scores can be found on the website of the CARLA autonomous driving leaderboard (https://leaderboard.carla.org/). If the penalty coefficient for infraction j is p_j, the infraction score is calculated as:

$$\text{IS} = \prod_{j}^{\text{Ped,Veh,Stat,Red}} (p^j)^{\#\text{infractions}^j}$$

- **Driving Score**: This is the main metric, which is calulated as the product of Route Completion and Infraction Score. If R_i is the percentage of completion of the i_{th} route and P_i is the infraction penalty on the i^{th} route, driving score is

calculated as:

$$\text{DS} = \frac{1}{N}\sum_{i}^{N} R_i P_i$$

5.3 Results

5.3.1 Comparing AIM-CNN and LatentTransFuser

The AIM-CNN baseline performs better than Latent Transfuser (LTF) in terms of both open-loop and closed-loop metrics.

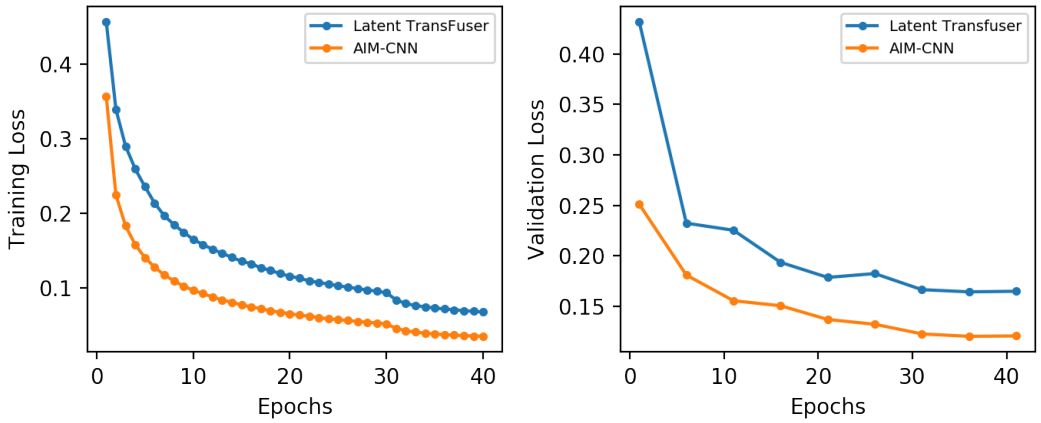

Figure 5.1: A comparison of AIM-CNN and LTF in terms of open loop metrics

We re-train LTF with only an L_1 loss between predicted and ground truth waypoints (removing all other auxiliary tasks/losses) with RegNetY-032 [39] (default used in [15]) architecture as the vision backbone. This is then compared with the AIM-CNN baseline using the same RegNetY-032 as the backbone. The vision backbones in both experiments are initialized with ImageNet pre-trained weights, as this is the default setting used for LTF in [15]. As can be seen in Figure 5.1, AIM-CNN reaches a better training and validation loss than LTF. It also outperforms LTF on closed-loop driving on the LAV routes by a large margin (Table 5.1). We hypothesize that this is due to the removal of all auxiliary losses, resulting in the BEV branch not playing a role anymore and simply overparameterizing the network. We therefore set AIM-CNN as the baseline for all experiments further on.

5.3.2 Training ViTs from scratch

Directly training Vision Transformers (ViTs) from scratch does not work and leads to severe overfitting.

Model	Encoder	Initialization	DS ↑	RC ↑	IS ↑
AIM-CNN	RegnetY-32	ImageNet	**26.29**	**67.14**	**0.49**
LTF	RegnetY-32	ImageNet	14.96	61.64	0.44

Table 5.1: AIM-CNN outperforms LTF in terms of driving scores

We train AIM-ViT with ViT-{Tiny, Small, Base} variants and observe that vanilla off-the-shelf vision transformers lead to overfitting. This can clearly be seen in Figure 5.2, where although the training loss of AIM-CNN is nearly the same as ViTs, there is a big difference in validation losses.

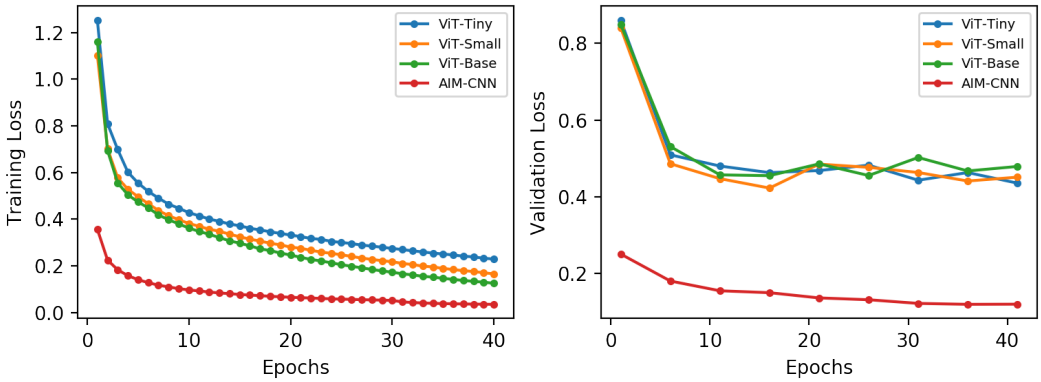

Figure 5.2: Training ViTs from scratch leads to overfitting.

5.3.3 Training ViTs initialized with ImageNet weights

ImageNet initialization is crucial for training vision transformer based image encoders. However, they still are not good driving agents and do not outperform CNNs.

Following the result in Section 5.3.2, we investigated what makes LTF and AIM-CNN work so well. We noted that the authors of TransFuser [15] used ImageNet initialized CNNs as the image encoder. For a fair comparison, we re-trained LTF and AIM-CNN with image encoders initialized from scratch and discovered that the performance of both drop significantly. We show this for AIM-CNN in Figure 5.3.

We then trained AIM-ViT initialized with ImageNet weights, and observed that they perform much better on the same expert data, and perform close to AIM-CNN in terms of open-loop metrics. Figure 5.4 shows this. However, these models are still not good driving agents, obtaining driving scores much worse than AIM-CNN (Table 5.2).

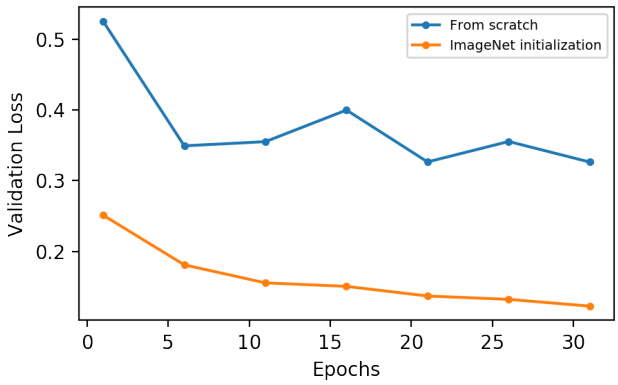

Figure 5.3: AIM-CNN requires ImageNet initialization for good performance.

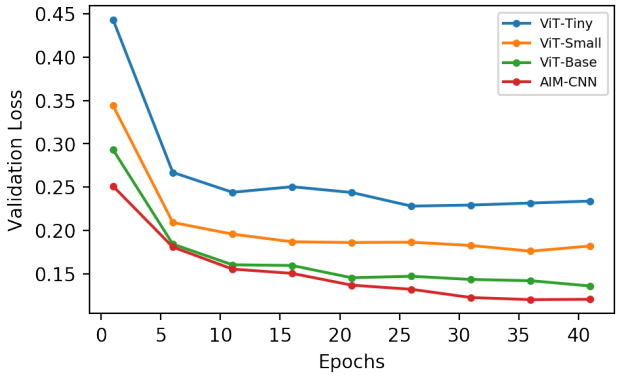

Figure 5.4: AIM-ViT also requires ImageNet initialization to achieve good open-loop scores.

Model	Encoder	Initialization	DS ↑	RC ↑	IS ↑
AIM-CNN	RegNetY-032	ImageNet	**26.29**	**67.14**	**0.49**
AIM-ViT	ViT-Tiny	ImageNet	10.89	50.74	0.29
AIM-ViT	ViT-Small	ImageNet	8.26	50.66	0.26
AIM-ViT	ViT-Base	ImageNet	5.60	41.942	0.30

Table 5.2: AIM-CNN outperforms AIM-ViT in closed-loop evaluation.

5.3.4 Replacing ViTs with Swin Transformers

ImageNet initialized Swin Transformer based image encoders are also not good driving agents.

The fact that AIM-CNN is the best driving agent until now implies that CNNs are much better than ViTs to optimize driving behavior. This prompted us to explore Swin Transformer as an image encoder, as it has a few properties of CNNs baked

in it's architecture (shifted window attention and hierarchical feature maps). We therefore experimented with AIM-Swin-{Tiny, Small, Base} variants initialized with ImageNet weights. We find that even Swin Transformer as a backbone does not result in representations optimized for driving. Figure 5.5 shows the results on open-loop metrics for this experiment. AIM-Swin matches AIM-CNN in terms of open-loop metrics.

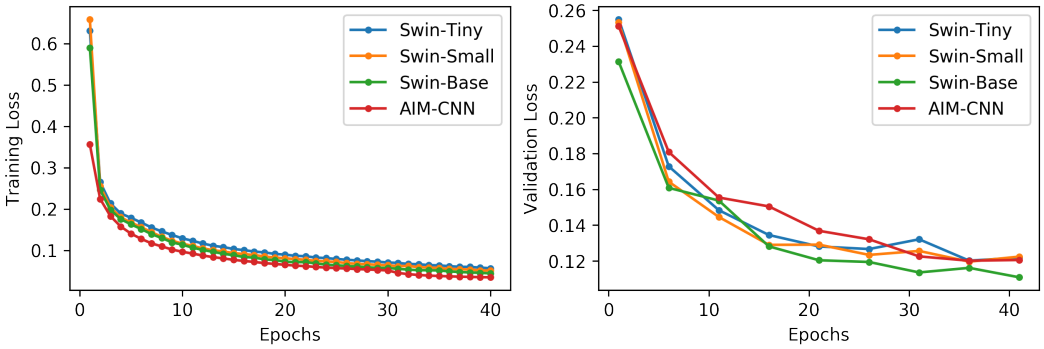

Figure 5.5: Comparing AIM-Swin with AIM-CNN

However, the closed-loop driving behavior is still worse in comparison to AIM-CNN (Table 5.3). Both ViTs and Swin Transformers cannot learn representations relevant for driving and perform poorly in closed-loop evaluation. Figure 5.6 compares various versions of AIM-ViT and AIM-Swin with AIM-CNN, and illustrates the result.

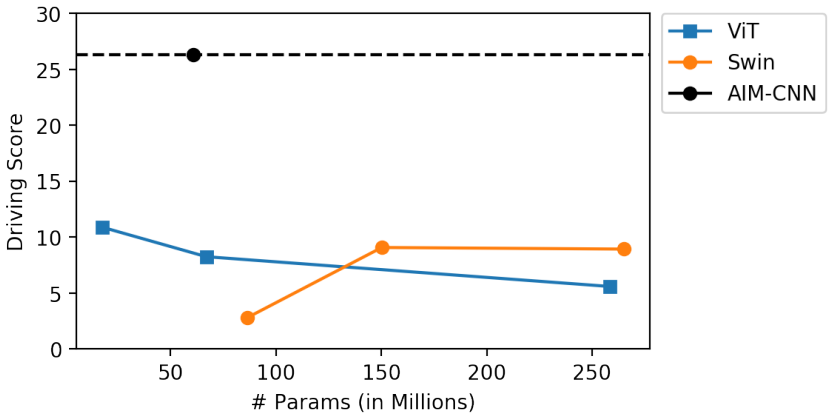

Figure 5.6: Both AIM-ViT and AIM-Swin perform poorly in comparison to AIM-CNN.

Model	Encoder	Initialization	DS ↑	RC ↑	IS ↑
AIM-CNN	RegNetY-032	ImageNet	**26.29**	**67.14**	**0.49**
AIM-Swin	Swin-Tiny	ImageNet	2.796	12.50	0.629
AIM-Swin	Swin-Small	ImageNet	9.082	51.620	0.248
AIM-Swin	Swin-Base	ImageNet	8.943	59.818	0.204

Table 5.3: AIM-CNN outperforms AIM-Swin in closed-loop evaluation.

5.3.5 ViTs initialized with MAE weights

ViTs initialized with MAE [26] weights perform better than ImageNet initialized weights, but are still sub-par to CNNs.

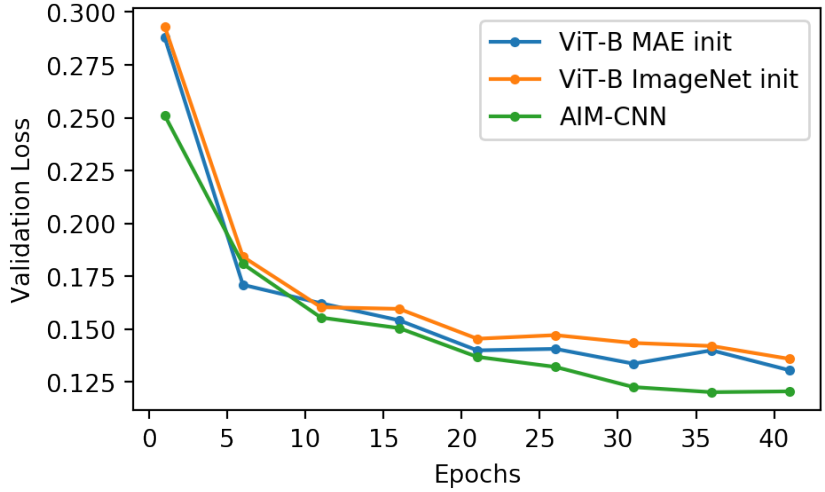

Figure 5.7: MAE initialization outperforms ImageNet initialization on open-loop metrics.

All experiments until now suggest transformer based image encoders do not learn representations that optimize driving behavior. However, Xiao et al. [55] and Radosavovic et al. [40] have shown that using ViTs as image encoders pre-trained using an MAE strategy result in excellent reinforcement learning policies for motor control. Such policies outperform supervised training from scratch and also ImageNet initialization. As autonomous driving can also be posed as a control task, we investigated if such ViTs also result in good driving agents. We train AIM-ViT-Base, with MAE initialized weights provided by the authors and call this AIM-ViT-Base-MAE. Contrary to [55, 40], we also need to train the encoder, as we find that freezing the encoder results in under-fitting. We hypothesize that this is the case due to the decoder being extremely simple.

Although not a better driving agent that AIM-CNN (Table 5.4), we find that MAE initialization outperforms ImageNet initialization, consequently outperforming

supervised training from scratch as well (Figure 5.7). However, PPGeo [54] reports such pre-training methods to not be as effective as ImageNet initialization for driving. However, it must be noted that there are a few key differences in our experimental setting:

- They make such claims for CNNs.
- Their training data is not as much as ours (40k frames v/s 168k frames).
- They evaluate on the Town05-Long routes, which is not as challenging as the routes we test on (LAV routes).

In contrast, we observe that this picture changes in the case of ViTs, suggesting traditional pixel-level pre-training strategies to be more effective than ImageNet initialization for autonomous driving.

Model	Encoder	Initialization	DS ↑	RC ↑	IS ↑
AIM-CNN	RegNetY-32	ImageNet	**26.29**	**67.14**	**0.49**
AIM-ViT	ViT-Base	MAE	19.15	60.51	0.39

Table 5.4: MAE pretraining is still not enough for ViTs to outperform CNNs.

5.3.6 ViTs initialized with MultiMAE weights

ViTs with MultiMAE [5] initialization perform worse in comparison to MAE initialization. Additional pre-training on CARLA data does not help as well.

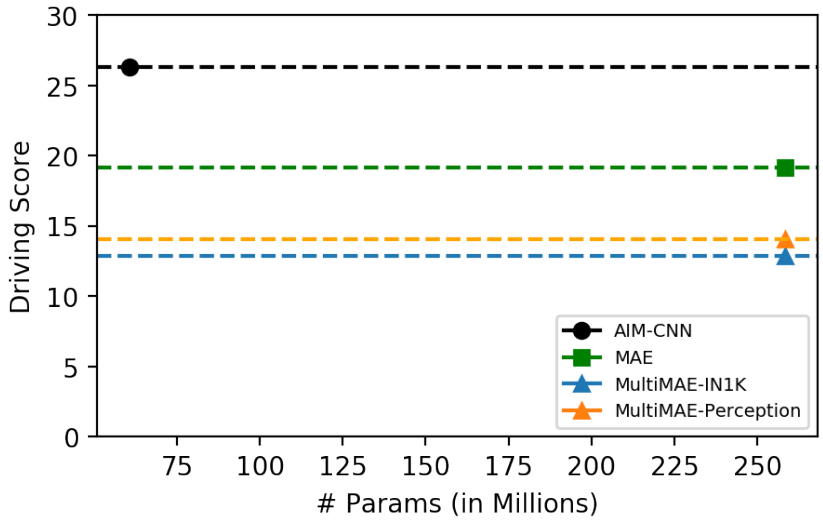

Figure 5.8: Comparison of MultiMAE and MAE initialization on closed-loop driving.

MAE initialization achieves the best driving performance among all transformer based encoders we have experimented with. MultiMAE [5] has shown to not only be as good as MAE on RGB modalities, but also achieve better transfer on depth estimation and semantic segmentation tasks. This suggests MultiMAE pre-training might result in better driving performance also as knowledge of depth and semantic modalities are important for taking driving decisions. We therefore trained AIM-ViT-Base, with MultiMAE initialized weights provided by the authors and call this initialization MultiMAE-IN1K. However, this kind of pre-training results in decreased driving performance in comparison to MAE initialization.

Model	Encoder	Initialization	DS ↑	RC ↑	IS ↑
AIM-CNN	RegNetY-32	ImageNet	**26.29**	**67.14**	**0.49**
AIM-ViT	ViT-Base	MultiMAE-IN1K	14.11	42.39	0.46
AIM-ViT	ViT-Base	MultiMAE-Perception	12.70	44.15	0.42

Table 5.5: Closed-loop evaluation of ViT-Base with MultiMAE initialization.

As ImageNet and CARLA have different depth values and semantic classes, pre-training on CARLA data can help the model adapt to the CARLA domain. We do so using the strategy described in Section 4.3.2 and call this initialization MultiMAE-Perception. We find that this also does not help and the driving performance is nearly the same. This is illustrated in Figure 5.8, and Table 5.5 shows the closed-loop scores of the same.

5.3.7 Qualitative Results (Waypoint Visualizations)

In this section, we present qualitative visualizations of a few success and failure cases of the best performing models i.e., AIM-ViT-MAE and AIM-CNN. Figure 5.9 is a simple scenario where the agent simply needs to drive straight. As can be seen clearly, both AIM-ViT-MAE and AIM-CNN can fit the waypoints and learn to drive in such situations properly.

Figure 5.10 shows the waypoint visualization of a scenario where the agent needs to turn right. As can be seen, AIM-CNN fits such situations properly. However, the AIM-ViT-MAE is biased towards the target point. This suggests that the overparameterized ViTs are learning shortcuts towards the target point.

We also explore scenarios where both the agents fail to perform. We discover that it is usually the scenarios where the agent needs to brake and come to a stop. Below, we depict a visualization of one of the cases. Figure 5.11 shows a scenario where the agent needs to come to a halting stop. AIM-ViT-MAE is not able to fit such scenarios and predicts to accelerate whereas AIM-CNN, although not fully correct in its prediction, is very conservative and predicts very little acceleration.

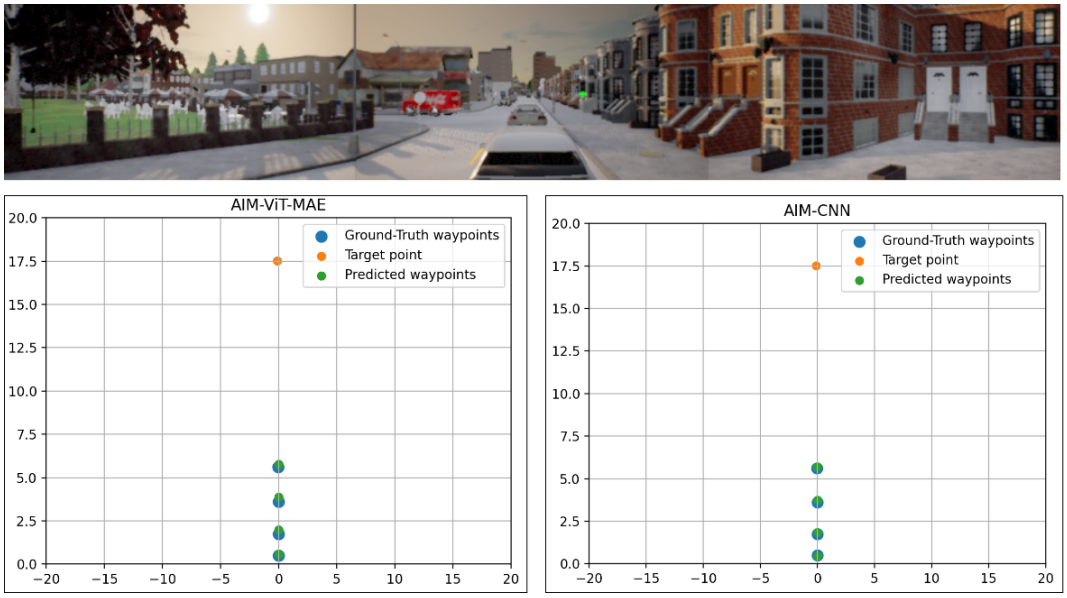

Figure 5.9: Predictions of AIM-ViT-MAE and AIM-CNN on a scenario when the agent simply needs to drive straight.

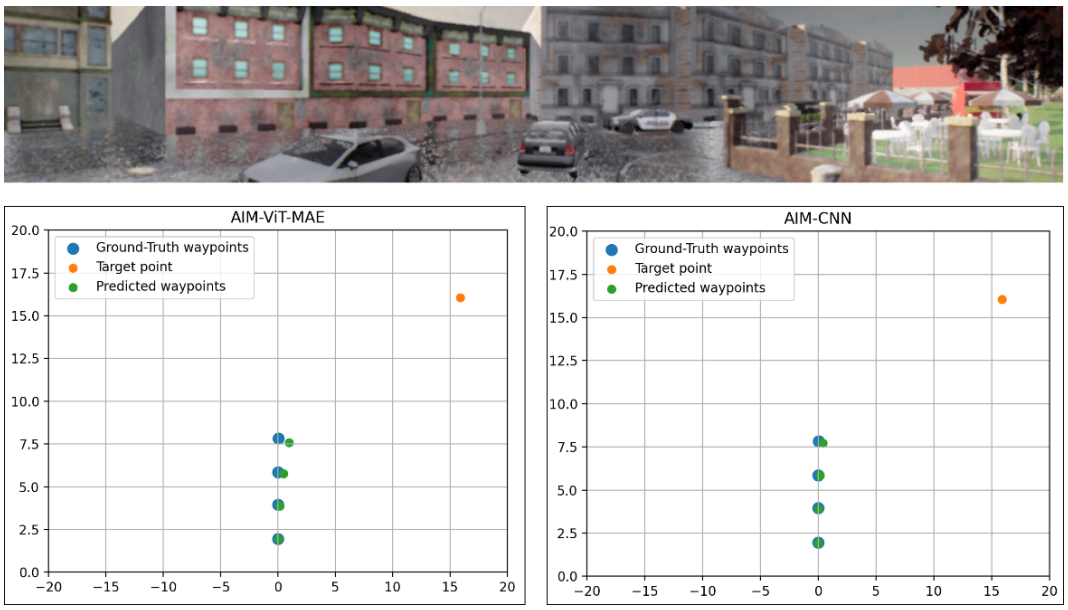

Figure 5.10: Predictions of AIM-ViT-MAE and AIM-CNN on a scenario when the agent needs to turn right.

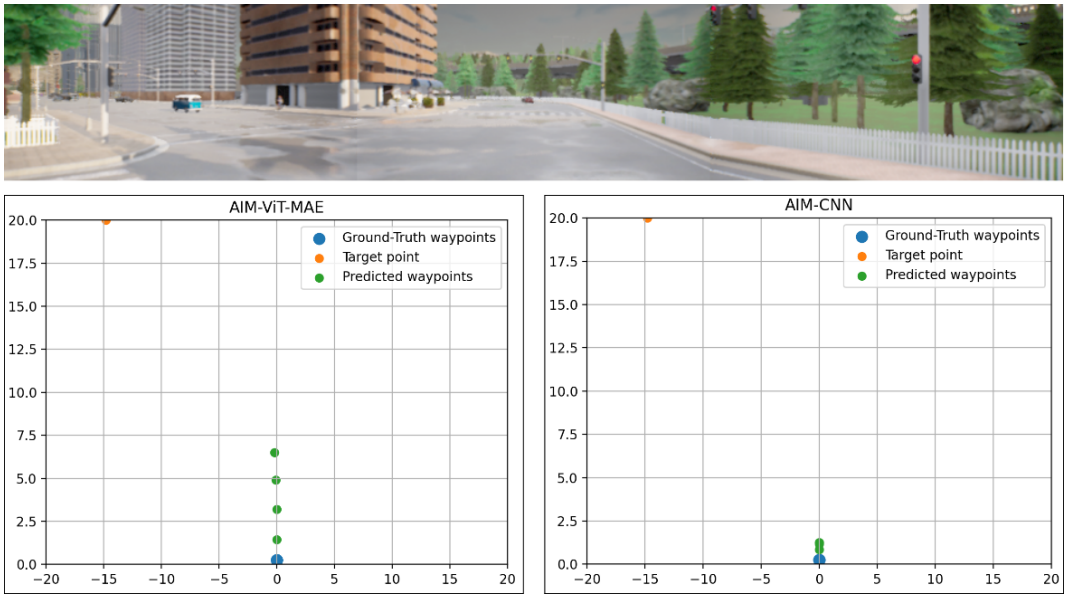

Figure 5.11: Predictions of AIM-ViT-MAE and AIM-CNN on a scenario when the agent needs to come to a halting stop.

5.3.8 Summary of obtained results

Although transformer based architectures reach open-loop scores near to AIM-CNN, it is clear that there is a big gap in closed-loop driving. Regarding transformers initialized with ImageNet weights, we hypothesize that because of the already encoded inductive biases in CNNs, they can focus on learning representations relevant to optimize driving, whereas transformers need to learn everything from scratch. The fact that ImageNet initialization helps all models converge supports this hypothesis. In addition, MAE, which already has captured more general image representations as a consequence of pre-training, can focus on learning representations for driving even more, resulting in better open-loop and closed-loop performance than ImageNet initialized models.

Regarding MultiMAE performing worse than MAE, we hypothesize that this could be attributed to the fact that MAE is pre-trained on RGB modality and then fine-tuned for end-to-end driving on the same modality. However, for MultiMAE, pre-training happens on RGB, depth and semseg modalities leading to learning entangled representations, but fine-tuning happens only on RGB modality. One reason could also be that more complex decoders are needed for better MultiMAE transfer. For example, the authors of MultiMAE use 4 ConvNext [32] blocks for semantic segmentation, thereby achieving better results than MAE, while we use a simple 4-layer GRU decoder for our task.

For greater confidence in our results, we evaluated the relevant best-performing

models on 3 seeds. Table 5.6 shows the mean and standard deviation across the runs, and also the L1 open-loop metric. Our claims above still stay consistent.

Model	Encoder	Initialization	$L_1 \downarrow$	DS ↑	RC ↑	IS ↑
AIM-CNN	RegNetY-32	ImageNet	**0.12**	**21.68 ± 3.28**	**63.96 ± 3.15**	0.40 ± 0.06
LTF	RegNetY-32	ImageNet	0.17	12.97 ± 1.86	54.40 ± 6.62	0.45 ± 0.06
AIM-ViT	ViT-Base	MAE	0.13	16.92 ± 2.36	57.95 ± 2.52	0.37 ± 0.02
AIM-ViT	ViT-Base	MultiMAE-IN1K	0.14	11.97 ± 2.16	46.46 ± 8.64	0.39 ± 0.05
AIM-ViT	ViT-Base	MultiMAE-Perc.	0.14	11.60 ± 0.88	43.27 ± 2.51	**0.45 ± 0.03**

Table 5.6: Relevant models evaluated on closed-loop driving over 3 seeds.

6 Conclusion

Current imitation learning approaches for autonomous driving encode images through a CNN. However, ViTs have shown to outperform CNNs in a lot of computer vision tasks and offer a more powerful backbone for transfer learning or feature extraction. This thesis therefore experiments if ViTs can be a better vision backbone for end-to-end autonomous driving. We find that training ViTs from scratch suffers from severe overfitting, and ImageNet initialization is crucial for such architectures to perform well in terms of open loop metrics. However, the representations learnt do not lead to better driving behavior in comparison to CNNs. We also find that the ViTs initialized with MAE weights work better than training from scratch and even ImageNet initialization. However, adding extra pre-training modalities does not help.

A limitation of our proposed approach is that we test such encoders in a stand-alone setting. It is possible that such backbones, when included as a part of a bigger, more sophisticated architecture perform differently than they do currently. This opens up a direction of future work – incorporating ViTs directly in existing architectures, for example TransFuser [15]. It is certainly possible that auxiliary tasks in TransFuser help ViTs provide better training signals to capture decisions relevant to driving.

Another limitation of our work is that the self-supervised methods we have experimented with, have shown to perform much better when pre-trained on in-domain data [55]. In our experiments, we have initialized weights pre-trained on the IN-1K dataset. An interesting direction from here could be to pre-train ViTs directly on CARLA data, which could potentially result in providing better representations. Also, our experiments on MAE were only possible on ViTs as the authors of MAE provide weights only for ViTs. Another direction of work stemming from here is to pre-train Swin transformers on IN-1K and test their transfer performance for autonomous driving.

Pre-training ViT architectures to specifically optimize driving behavior is another interesting direction for future work. Visuomotor policy pre-training [57, 54, 58] show that such pre-training strategies are effective for autonomous driving. However, all prior work shows this for CNNs. It could be the case that such pre-training strategies make ViTs learn representations relevant for driving.

Bibliography

[1] Automated vehicles for safety.

[2] Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. https://www.sae.org/standards/content/j3016_202104/.

[3] The future of driving in the united states, Mar 2021.

[4] Jay Alammar. The illustrated transformer.

[5] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. MultiMAE: Multi-modal multi-task masked autoencoders. *arXiv preprint arXiv:2204.01678*, 2022.

[6] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *ArXiv*, abs/1812.03079, 2019.

[7] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020.

[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[10] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to drive from a world on rails. In *ICCV*, 2021.

[11] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *CVPR*, 2022.

[12] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning (CoRL)*, 2019.

[13] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation, 2022.

[14] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *International Conference on Computer Vision (ICCV)*, 2021.

[15] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[16] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 1–9. IEEE Press, 2018.

[17] Felipe Codevilla, Eder Santana, Antonio López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. pages 9328–9337, 10 2019.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[19] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[21] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[22] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations, 2021.

[23] Ainaz Eftekhar, Alexander Sax, Roman Bachmann, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans, 2021.

[24] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

[25] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao. A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(01):87–110, jan 2023.

[26] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s):1–41, jan 2022.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[31] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[32] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[33] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. *Junior: The Stanford Entry in the Urban Challenge*, pages 91–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[34] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. In *Conference on Robot Learning*, 2018.

[35] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann Cun. Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[36] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting, 2016.

[37] Dean A. Pomerleau. *ALVINN: An Autonomous Land Vehicle in a Neural Network*,

page 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[38] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[39] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces, 2020.

[40] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022.

[41] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[42] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[43] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, A. Sophia Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. In *Conference on Robotic Learning (CoRL)*, 2022.

[44] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.

[45] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Conference on Robot Learning (CoRL)*, 2018.

[46] Hao Shao, Letian Wang, RuoBing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. *arXiv preprint arXiv:2207.14024*, 2022.

[47] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era, 2017.

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021.

[49] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner,

M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. *Autonomous Driving in Urban Environments: Boss and the Urban Challenge*, pages 1–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[50] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[52] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos, 2015.

[53] Wikipedia. Cruise (autonomous vehicle) — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Cruise%20(autonomous%20vehicle)&oldid=1152208009`, 2023. [Online; accessed 03-May-2023].

[54] Penghao Wu, Li Chen, Hongyang Li, Xiaosong Jia, Junchi Yan, and Yu Qiao. Policy pre-training for autonomous driving via self-supervised geometric modeling. In *International Conference on Learning Representations*, 2023.

[55] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv:2203.06173*, 2022.

[56] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, 2021.

[57] Jimuyang Zhang, Ruizhao Zhu, and Eshed Ohn-Bar. Selfd: Self-learning large-scale driving policies from the web. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17295–17305, 2022.

[58] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. *European Conference on Computer Vision (ECCV)*, 2022.

[59] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. Do rnn and lstm have long memory? In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.