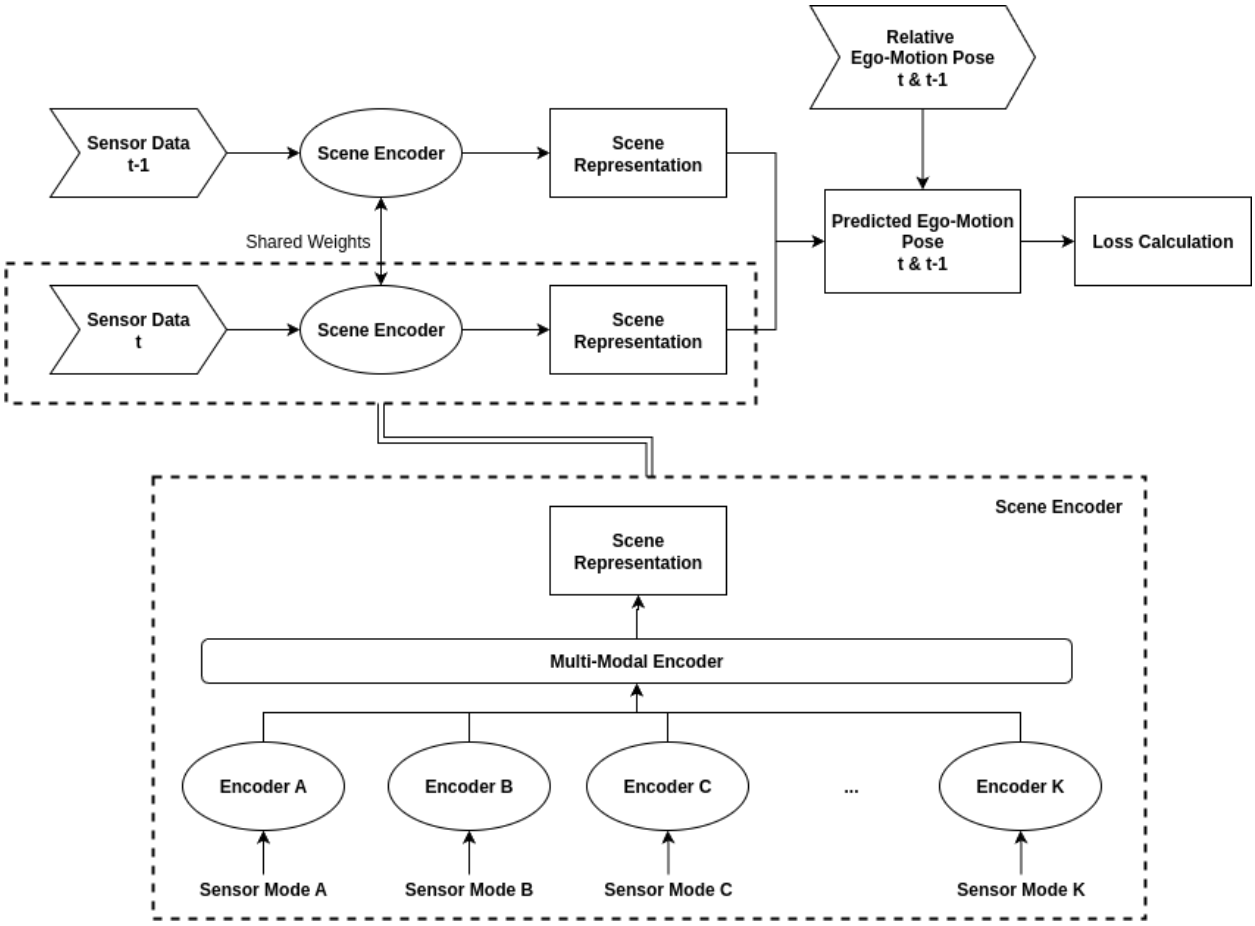
CHALMERS
UNIVERSITY OF TECHNOLOGY



Annotation-free Learning for Sensor Fusion in ADAS

Multi-modal Pre-training with Ego-vehicle States

Master's Thesis in Information and Communication Technology

Maria Björkman
Ludvig Tvingby

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis project report 2025

Annotation-free Learning for Sensor Fusion in ADAS

Multi-modal Pre-training with Ego-vehicle States

MARIA BJÖRKMAN
LUDVIG TVINGBY

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Annotation-free Learning for Sensor Fusion in ADAS
Multi-modal Pre-training with Ego-vehicle States
Maria Björkman
Ludvig Tvingby

Supervisor: Tzu-Jui Wang & Maria Priisalu, Zenseact AB
Examiner: Lars Hammarstrand, Electrical Engineering (E2)

Master's Thesis 2025
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: The network architecture design of multi-modal pre-training utilising ego-vehicle states.

Typeset in LaTeX
Gothenburg, Sweden 2025

Annotation-free Learning for Sensor Fusion in ADAS
Multi-modal Pre-training with Ego-vehicle States
MARIA BJÖRKMAN
LUDVIG TVINGBY
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Vehicle automation has the potential to significantly improve road safety. Achieving comprehensive vehicle perception requires systems that optimally combine information from multiple sensor modalities. Such systems leverage the strengths of each modality while compensating for their weaknesses. By continuously encoding and fusing information from cameras, LiDARs, RADARs and the motion of the ego-vehicle, a dynamic representation of the surrounding environment can be created and maintained.

A major challenge for these systems is the large amount of annotated data required for training, as manual labelling creates a significant bottleneck for scalability. In this study, a pre-training task for a multi-modal machine learning model was implemented and evaluated. To circumvent labour-intensive labelling, self-supervision was employed, with both the model input and the supervision signal involving annotation-free data. The pre-training aimed to learn general features related to sensor pose changes by predicting ego-vehicle pose changes using odometry data.

To assess pre-training performance, the features were then used as initial weights for fine-tuning a perception model. The performance of the perception model using baseline weights trained on annotated data was similar to that using weights trained on annotation-free data, indicating that the proposed method is viable. However, further testing is required to establish statistical significance. Future work could explore implementing attention-based methods for feature matching between scene representations to improve model performance.

Keywords: ADAS, Annotation-free, Ego-vehicle, Multi-modal, Perception, Pre-training, Sensor Fusion, Transformer.

Acknowledgements

This master's thesis could not have been completed without the support of our supervisors from Zenseact, Tzu-Jui Wang and Maria Priisalu. They supported us through frequent meetings and by swiftly answering our questions when we felt stuck and provided clarity throughout the entitre process. We also thank our examiner, Lars Hammarstrand, especially for putting together the "Master thesis school". This grouped thesis students together to help us with various topics throughout the progression of the thesis and grounded the work to the academic requirements. Lastly, we want to thank friends and family who supported us along the way unwaveringly, helping us persevere through the challenges we were facing.

Maria Björkman & Ludvig Tvingby, Gothenburg, November, 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in the order they are first mentioned:

ADAS	Advanced Driver Assistance Systems
LiDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
AI	Artificial Intelligence
NN	Neural Network
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
AD	Autonomous Driving
FNN	Feed Forward Neural Network
MLP	Multi-Layer Perceptrons
DML	Deep Machine Learning
NLP	Natural Language Processing
Adam	Adaptive Moment Estimation
SGD	Stochastic Gradient Descent
ZOD	Zenseact Open Dataset
DETR	DEtection TRansformer
ReLU	Rectified Linear Unit
MAE	Mean Absolute Error
GNSS	Global Navigation Satellite Systems
IMU	Inertial Measurement Unit
GASP	Unifying Geometric and Semantic Self-Supervised Pre-training for Autonomous Driving
UnO	Unsupervised Occupancy Fields for Perception and Forecasting
OD	Object Detection
HT	Holistic Trajectory
AP	Average Precision
ATE	Average Translation Error
ASE	Average Scale Error
AOE	Average Orientation Error
ACE	Attribute Classification Error
IDS	Identity Switch
ZDS	Zero Detection Score
TDE	Trajectory Displacement Error

Contents

List of Figures

List of Tables

1
Introduction

Road traffic accidents are a global public health crisis, claiming approximately 1.19 million lives each year and leaving 20 to 50 million more people with non-fatal injuries, which often result in long-term disabilities. Beyond the human toll, these accidents impose significant economic burdens on individuals, families and nations, accounting for an estimated 30% of the gross domestic product (GDP) in many countries [1]. In recognition of this, the European Union has committed to the goal of "Vision Zero" for road safety, of which a pivotal role in order to achieve this is safe vehicles. With the integration of advanced technologies, accidents can be prevented and the severity of crashes can be minimized by combining safety systems, such as intelligent sensors, real-time monitoring and dynamic systems. These systems detect and react to potential hazard before they lead to accidents. These systems become increasingly more crucial as vehicle automation advances and human-vehicle interactions become more integrated into the society [2].

Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS) represents a significant leap in modern automotive technology. A fundamental part of realising these systems is acquiring real-time information of the environment through multiple heterogeneous sensors. With inputs such as cameras, Light Detection and Ranging (LiDAR) and Radio Detection and Ranging (RADAR), it is of key importance to utilize the full potential of the multi-modal sensor data in order for the vehicle to comprehensively perceive the driving environment. Different sensor types display different advantages and disadvantages, which make the multi-modality of the sensor data of great interest [3].

In order for a vehicle to achieve cognitive capabilities such as interpret complex and dynamic traffic scenarios, trajectory planning, efficient navigation, ADAS leverages sophisticated algorithms which transform sensor inputs into actionable insights. A fundamental technique to realise this is sensor fusion. Sensor fusion is the process of integrating data from multiple sensors and reducing the uncertainty of the resulting information. A key challenge for the practical implementations of these operations in real-time for ADAS is the computational complexity of processing the large amount of raw sensor data. This complexity in processing sensor data is a reason why more advanced machine learning techniques can play a valuable role in ADAS [3].

The thesis aims at exploring scalable multi-modal sensor fusion through unsupervised learning. The project focuses on pre-training models that reason based on multiple heterogenous sensors. These pre-trained models can then be fine-tuned for

different downstream tasks. Furthermore, it is crucial for such a model to learn robust multi-modal representations to improve tasks such as Object Detection (OD) and Holistic Trajectory (HT), as all such tasks requires a reliable perception of the environment surrounding the vehicle. Lastly, it is of interest to reduce the dependency on annotated datasets during model learning.

1.1 Purpose and Research Questions

In order for an autonomous vehicle to perceive its surroundings, it should be able to process and contextualize data from many sensor inputs of different modalities. Therefore, a model which needs to be able to perform at downstream tasks, such as OD, should perform well with multi-modal inputs. One way of modelling such cross-modal interactions is through an attention mechanism described in [4]. However, attention operations are parameter intensive on the order of tens of millions in order to model complex interactions as well as dealing with the cross-modal domains. This leads to a slow convergence rate for any downstream ADAS tasks during training. The overarching goal is therefore to utilise pre-training where a model is initialized on an independent task which has some similarity to the downstream task of interest. A successful pre-training could involve a scalable way of including much more training samples as well as using samples with minimal manual annotations involved. Furthermore, most of the proposed pre-training methods today, such as the state of the art model Unsupervised Occupancy Fields for Perception and Forecasting (UnO) [5], have a focus on a single modality.

The aim of this thesis is to develop an annotation-free pre-training approach for sensor fusion in ADAS with multi-modal sensor input. The idea is to take advantage of the inertial signals from the navigational system already present in the vehicle, which means no additional cost. The inertial data is annotation-free and present in all standard vehicles and when using it in the proposed approach it is computationally scalable and the sensors are not required to be synchronized. The expected outcomes are: (1) an implementation of an annotation-free pre-training method and (2) an evaluation of its effectiveness in downstream tasks.

Research questions:

- Earlier attempts at pre-training with ego-motion states as detailed in [6] had issues with distinguishing between static and dynamic objects when only using data from cameras. Does the adaption to a multi-modal context which includes cameras, LiDARs and RADARs make up for this?
- Do the features pre-trained on relative sensor poses correlate well with downstream tasks such as OD?
- Does the model acquire an object-level understanding such as learning where objects and lanes are?
- Do the curated datasets include diverse enough driving scenarios?

- Is the proposed method scalable with large datasets?

1.2 Limitations and Demarcations

The scope of the project focuses on the development and implementation of a proof of concept to validate model feasibility. The performance evaluation will focus on the downstream tasks of OD and trajectory planning. Furthermore, the sensor placement for the collection of ego-motion poses is assumed to be fixed on the ego-vehicle, while the fields of view of the other sensors, which are used for scene information, are assumed to be partially overlapped.

- This project will primarily utilise an internal Zenseact dataset for training, which is similar to the Zenseact Open Dataset (ZOD) with expanded sensors [7]. The considered dataset includes sensor data for cameras, LiDAR and RADAR, as well as high precision Global Navigation Satellite Systems (GNSS) and Inertial Measurement Unit (IMU) information.
- The investigated unsupervised learning approach will capitalize on ego-motion utilised as a supervisory signal [6].
- The considered model architecture will include Perceiver [8], which excels at fusing information from multiple sensors.
- There is a pre-defined sensor setup in the dataset, however, the proposed method should handle all sensor modalities arbitrarily.
- The models and training will be implemented in the Python programming language and specifically the PyTorch package for deep learning.

1.3 Ethical and Sustainability Aspects

Sustainability and ethics play a crucial role in the development and implementation of unsupervised learning in ADAS. From a sustainability perspective, the project focuses on enhancing the efficacy and efficiency of autonomous driving systems. This would contribute to fewer road accidents, reduced fuel consumption, and emissions by improving safety and optimising vehicle performance. The work we aim to achieve well relates not only to the goal of Zenseact but that of the United Nations [9]. Creating tools that aid the car to manoeuvre more safely and responsibly takes the world closer to the set goal of zero accidents on the road.

By leveraging large amounts of unlabelled data, the proposed approach also reduces the need for manual labelling, which can be resource-intensive. Thereby decreases the overall environmental footprint of data collection and processing.

However, unsupervisedly training a model from these large multi-modal sensor data can potentially itself be computationally expensive. Thereby, this thesis project does not guarantee the training to drain fewer resources. Nevertheless, the proposed methodology could lead to a more effective model training process, translating to less energy consumption. The use of multi-modal sensor data allows for more accurate and inclusive decision-making, reducing the risks of errors that could arise from

relying on a single sensor type.

Moreover, it is essential to consider data privacy, especially when using cameras and sensors that may capture sensitive information about individuals. The project will aim to address these concerns by ensuring responsible data management practices, such as anonymising data where necessary and adhering to regulations surrounding data protection and privacy. For example, ZOD employs anonymisation techniques for personally identifiable information, including human faces and vehicle license plates [7]. Overall, the work in this thesis is aligned with ethical Artificial Intelligence (AI) principles, aiming to create a more sustainable, equitable, and safer future.

2
Theory

This chapter introduces the central core concepts of machine learning before focusing on the specific network architecture. Additionally, details of sensor modalities and using ego-motion for feature learning as a supervisory signal will be presented.

2.1 Machine Learning

Machine learning tasks can at a fundamental level be defined as how to process a collection of quantitively measured features of either an event or object. These feature collections are typically represented as a vector

$$\mathbf{x} \in \mathbb{R}^n$$

where every vector entry x_i in turn represents a different feature. In order to evaluate a machine learning algorithm a quantitive measure of performance is required. Standard examples are accuracy and error rate, however, depending on the task more specialized metrics must be designed in order to correspond with the desired system behaviour [10, ch. 5].

In machine learning, one can broadly categorize training into supervised and unsupervised learning. Supervised learning is the case where a model learns to match inputs to targets. This requires the model to be trained on a dataset with labelled data for both input features and targets. The learning objective is to equip a model with a fine generalization capability on a wide data spectrum. Supervised learning is widely used in problems such as classification and regression. In contrast, unsupervised learning is instead a technique where training is performed on unlabelled data. Unlike supervised learning, it is used to identify patterns and features without explicit guidance. That way, the learning algorithms explore underlying latent relationships among data samples. However, a disadvantage with unsupervised methods is that they are generally more computationally complex as well as requiring larger datasets. Data exploration, pattern recognition and data pre-processing are fields where unsupervised learning has multiple applications [11][12].

The distinction between supervised and unsupervised learning is however not rigidly defined. Instead unsupervised learning can informally be defined as attempts at retrieving information from a distribution which do not utilise manual work from humans in order to provide annotations [10].

One key area of machine learning is determining similarity. When comparing data of the same data type and domain, it is easy to compare two elements with purely mathematical metrics, such as the Euclidean or Cosine distance. A problem that is less easily defined as a mathematical problem with statistical evaluations is when comparing elements with different data types and domains. One solution to this is processing and projecting features into a hidden representation and compare the elements in this new representation [13].

2.1.1 Deep Machine Learning

A Neural Network (NN) is a machine learning method which uses simple neurons in layers, see fig. 2.1 and fig. 2.2, to build a pattern recognition model using weights and biases. Furthermore, Deep Machine Learning (DML) is a subset that is focused on using multiple layers of NN in order to execute tasks. A NN is built of layers of processing units that are called neurons. In the case of the simplest form of NN, called Feed Forward Neural Network (FNN), where each of the neurons in the first layer reads an input value, which is then multiplied with a weight and finally the result is then sent to each neuron in the next layer. A widely used implementation of this is the Multi-Layer Perceptrons (MLP) [13].

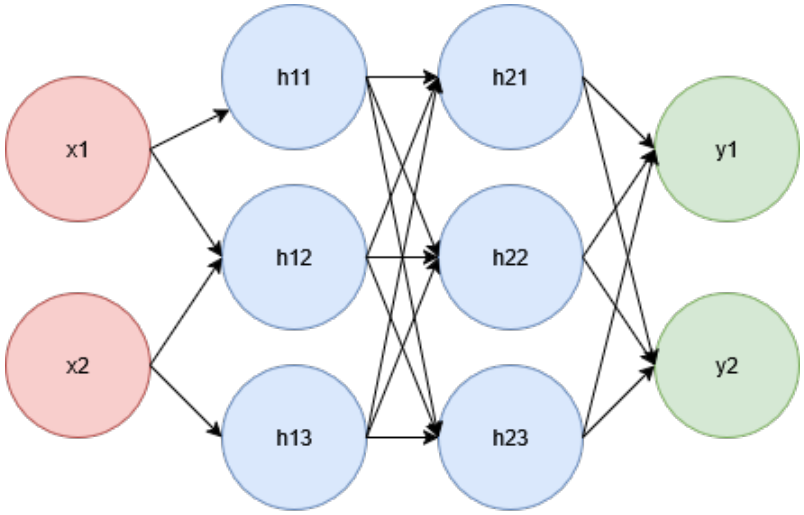


Figure 2.1: *A simple NN with two hidden layers. In the common case of a linear transformation, each node in the hidden layers can be defined according to eq. 2.2 where each arrow represents a weight.*

The fundamental goal of a MLP is to approximate a function $f(\mathbf{x})$ to match $f^*(\mathbf{x})$ by iterative learning, where the model compares true target values with predictions. However, a training dataset consists of many approximate and noisy samples of the target values, as such the training data is modelled as noisy samples of $f^*(\mathbf{x})$. These samples are what is used to directly specify what is required of the output layer by comparing to a predicted value. In contrast, the behaviour of the hidden layers is

not directly specified by the training data. Additionally, the layers of a NN can be represented as a chain structure

$$f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))) \tag{2.1}$$

where the depth of the model is defined by the number of layers used, while the width is determined by the dimensionality of each layer. The first layer of the network is represented by $f^{(1)}$, while the second is defined by $f^{(2)}$. The final layer is termed the output layer. In this case, the equation describes the same network architecture shown in fig. 2.1. With increased depth and parameter complexity, networks are defined to be within the domain of DML [10, ch. 6].

The architecture of a simple FNN, see fig. 2.1, consists of multiple fully connected layers where input data is represented by neurons in the input layer. The processing occur in the hidden layer(s) while the final prediction is represented with one or more neurons in the output layer. During learning, each neuron in the hidden layers of the network computes a weighted sum, z,

$$z = \sum_i \omega_i x_i + b \tag{2.2}$$

of the inputs together with the weights, ω, and biases, b. These weights and biases are adjusted during model optimisation. In order to increase model flexibility to model more complex data, it is possible to extend the depth and width of the network. Furthermore, a way to extend the linear model to represent non-linear behaviours is to apply a transformation on the input, $\phi(\mathbf{x})$. In this case the function ϕ represents the non-linear transformation [10, ch. 6].

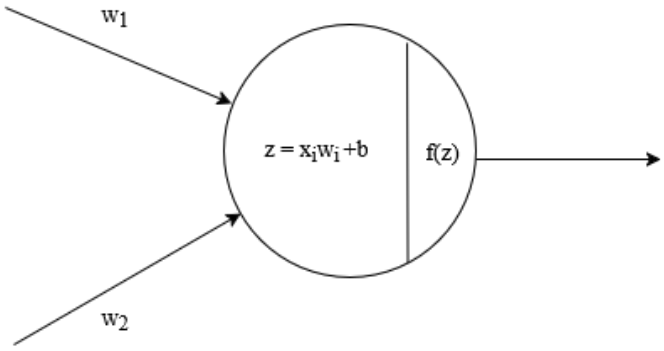


Figure 2.2: *An artificial neuron in a NN, which includes a weighted sum in the linear case as well as an activation function.*

In addition to the weighted sum of the input of a neuron, see fig. 2.2, also includes an activation function, $f(z)$, which transforms the output before passing it to the next layer. Two common activation functions are the Rectified Linear Unit (ReLU) and the sigmoid function. We can define ReLU as:

$$f(z) = \text{ReLU}(z) = \max(0, z) \tag{2.3}$$

which introduces non-linearity to the NN and is in many cases used in the neurons of the hidden layers. The sigmoid function

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

is often used as the activation for the output layer in the case of binary classification as it produces a value between one and zero.

The softmax function is a normalised exponential function

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.5}$$

and produces values in the interval $(0, 1]$. The function is often used in the case of multi-class classification tasks where it outputs a probability distribution.

2.1.2 Loss Function and Optimizer

What separates machine learning from a standard optimization problem is that the model must perform well on inputs it was not trained on. This ability is what is described as generalization. The performance of the predictions for a NN is defined by a loss function. As such, the choice of loss function is an important design decision which affects the learning process. Additionally, an optimizer is what defines how the network should change its weights and biases.

Learning consists of three phases; the forward pass, loss calculation and backwards propagation. The forward pass is when input data is passed through the hidden layers and an output is predicted. The loss calculation compares the prediction to the target value and produces a calculated loss. During the backwards propagation the calculated loss from the last step is sent backwards through the network. It is during this step that the weights and biases are adjusted.

The choice of loss function depends of the specific task the NN is designed for. A common loss is Mean Absolute Error (MAE), also named L_1 loss which only considers the magnitude of the absolute error. The MAE loss is calculated using the input vector $\mathbf{x}$, output vector $\mathbf{y}$ and the amount of predictions made, D.

$$\mathcal{L}_{L1}(x, y) = \frac{1}{D} \sum_{i=1}^{D} |x_i - y_i| \tag{2.6}$$

The optimizer defines how the loss is minimized and how the weights and biases are adjusted for increased performance. The loss is an important parameter in the training process as this determines how it learns and thereby how it converges. During training, gradients are calculated determining the magnitude and direction of the parameter adjustment with respect to the chosen loss function. A commonly used optimizer is the Stochastic Gradient Descent (SGD), which can be described

as gradient descent performed on a randomly chosen subset of data. This method essentially proposes a new point, $x^{'}$, according to

$$x^{'} = x - \epsilon \nabla f(x) \tag{2.7}$$

where the magnitude and direction of the learning process is determined using the gradient, ∇ of the function and compares this to the previous point, x. ϵ is the learning rate determining the magnitude of the step [10, ch. 5]. Adaptive Moment Estimation (Adam) is a variant which used momentum with moving averages of first and second moment as well as adaptive learning rates for separate parameters in order to make convergence faster.

It is also possible to choose the learning rate, which defines how often the magnitude is updated between iterations. If faster convergence is of interest, a parameter value of larger magnitude should be considered. On the other hand, a small value value slows down training but prevents overfitting. A large value for learning rate can cause overshooting and make the model unable to find the optimum solution. In contrast if a too small learning rate, close to zero, is chosen, this can instead cause slow convergence or get stuck in a local minima instead of the global minima.

What determines the performance of a machine learning algorithm is its ability to (1) predict outcomes of the training set with minimal error and (2) keeping the model generalised, which is when the model can handle new data with similar losses. This then corresponds to the concepts of underfitting and overfitting. Underfitting is defined as when the model is not achieving a low enough error value for the training data, while overfitting instead occur when the training loss is low but the validation loss is high, it means the model has difficulty adapting to new data and therefore does not generalise well. By changing the capacity of the model it is possible to control the likelihood of the model to underfit or overfit. A model with high capacity has a tendency to overfit by overly adapting to properties in the training set while a model with low capacity have difficulty adapting to the training data [10].

By comparing metrics involving ratios of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) evaluation of the performance becomes possible. Depending on the specific network task different metrics are of interest. Accuracy defines the overall correctness in the models prediction while precision describes the accuracy of the predictions. Accuracy and precision calculations are defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2.8}$$

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.9}$$

.

2.2 Transformer Architecture

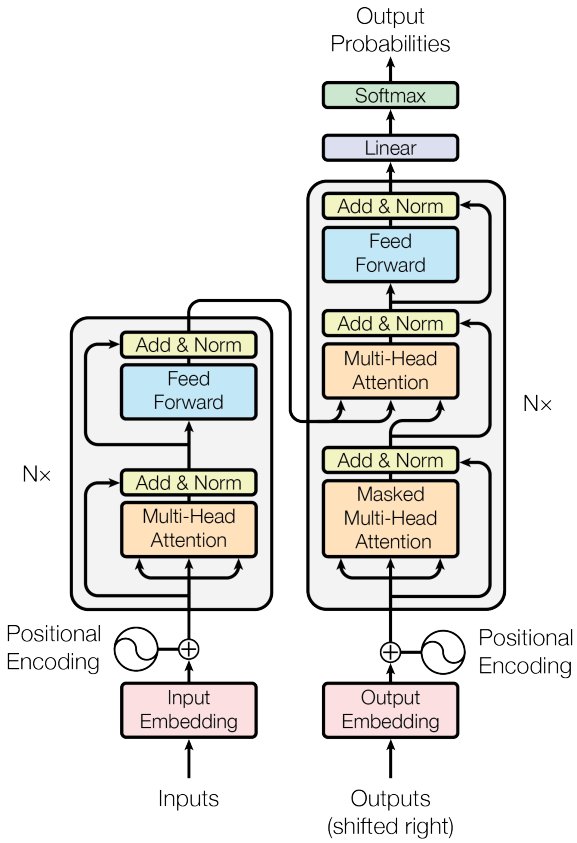
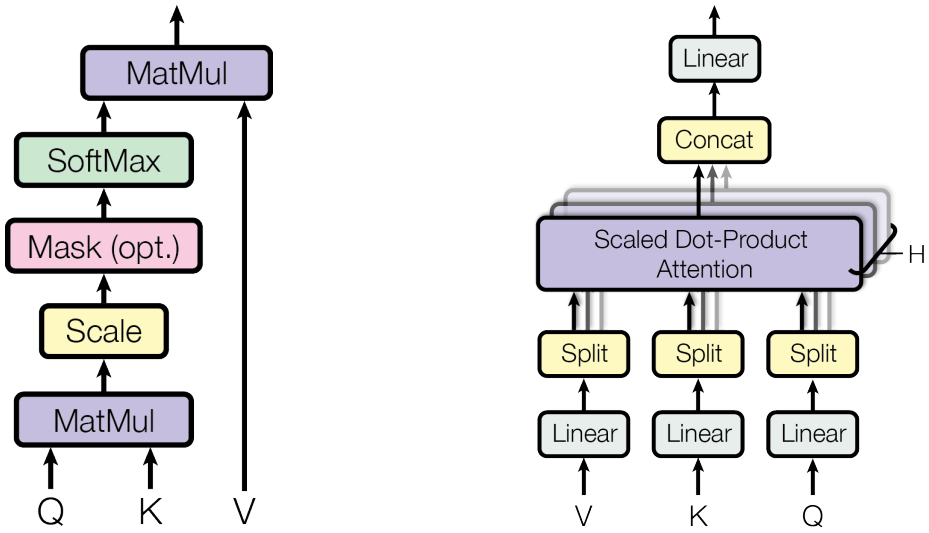


Figure 2.3: *The Transformer architecture from [4]. Input, output, and positional embeddings are processed through multi-headed attention, followed by a FNN, and finally a softmax layer produces the output probabilities.*

The Transformer is a network architecture which utilises attention mechanisms in order to model dependencies between input and output globally. Of networks used for transductive learning, the Transformer is the first to eschew sequence based Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) in order to compute input and output representations entirely based on attention mechanisms. A key point of this architecture is the ability to parallelize its computational operations [4].

A representation of the Transformer architecture is presented in fig. 2.3. The model has an encoder-decoder structure where the encoder is used to map a sequence of symbol representation to a sequence of continuous representations. In the decoder an output sequence is generated one symbol at a time. Each step also uses the previous symbol as input when producing the next symbol. The architecture uses stacked self-attention as well as fully-connected point-wise layers in both the encoder and the decoder [4].

2.2.1 Attention Mechanism



(a) *Scaled dot-product attention block from [4]. This block computes attention by taking the dot product of the query with all keys, scaling, applying a softmax and multiplying by the values.*

(b) *Multi-headed attention block from [4]. This block applies multiple scaled dot-product attention mechanisms in parallel, concatenates their outputs, and projects them to produce the final result.*

Figure 2.4: *Key components of the attention mechanism in Transformers are the scaled dot-product as well as the multi-headed attention block.*

The attention mechanism used in the Transformer architecture can at its most basic be a mapping function. A query and a set of key-value pairs is mapped to an output, then a weighted sum of all the values is computed with a compatibility function. This is a function of the query with its respective key. The attention used is the scaled dot-product attention, presented in fig. 2.4a. The input queries and keys have dimension d_k, while the values have dimension d_v. The dot product is computed for the query using all keys, then each product is divided by $\sqrt{d_k}$ and finally a softmax function is used to produce the weights. When implemented the attention function is computed on a set of queries simultaneously

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.10}$$

with the input formatted as a matrix, Q, as well as the keys and values in the form of matrices K and V [4].

An alternative to performing one attention function with keys, values and queries is to instead linearly project them with different linear projections. Attention is then performed on each projected version in parallel, which is then concatenated and projected again

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O. \tag{2.11}$$

in order to produce the final output values. This is called multi-headed attention and a visualization is presented in fig. 2.4b and can be computed through
where the attention block This implementation of the attention process makes it possible to attend information from different subspace representations at different positions [4].

As the model performs many computations in parallel without the use of recurrence or convolution it needs to keep the sequence order in memory. Thus positional encodings are added to the input embeddings and have the same dimensions as the input. This allows for summation of positions and input embeddings.

2.3 Perceiver Architecture

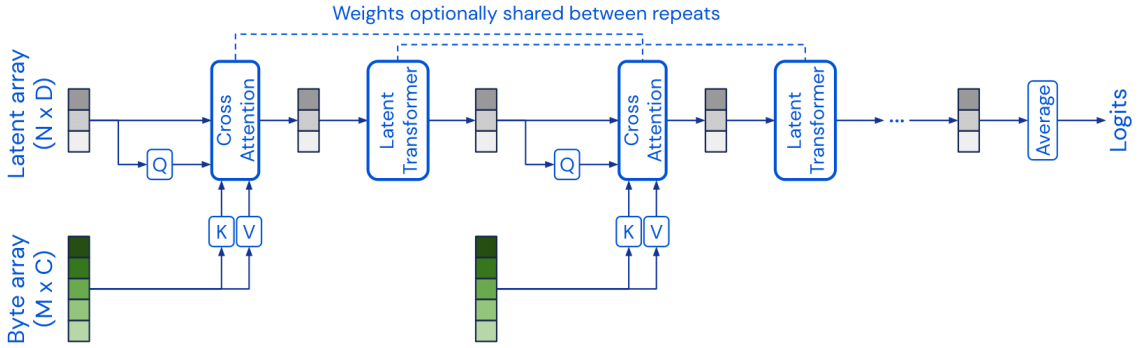


Figure 2.5: *Visualisation from [8] of the network architecture of the Perceiver model. This model utilises iterative cross-attention to scale high-dimensional input as well as handle arbitrary input without making domain assumptions.*

The Perceiver model builds on Transformers with the goal of handling arbitrary configurations of many different types of input modalities. Furthermore, unlike standard Transformers which scale quadratically with the inputs it decouples this computational scaling issue from the input size. The model is also designed to make few assumptions with regards to relationships between inputs, unlike many other methods which focus on dealing with one input modality. This is achieved by attending to inputs iteratively, where the model limits the capacity of the next iteration based on the most relevant inputs from the previous iteration with an asymmetric attention mechanism [8].

The architecture, as seen in fig. 2.5, is built on two larger components. A cross-attention module and a Transformer tower. The cross-attention maps a byte array and a latent array to a latent array while the Transformer tower maps a latent array to a latent array. The size of the byte array is dependant on the input data while the size of the latent array is a hyper-parameter and generally smaller. The model performs cross-attention and the Transformer in alternate.

In summary the model can also be described as performing an end-to-end clustering of its inputs while using the latent positions as the centre of the clusters.

2.4 Pre-training

The use of pre-training techniques for machine learning models have been present since close to the inception of machine learning [14, ch 2.1]. The benefit of pre-trained models now make it an essential part of machine learning models as they become increasingly large and complex, due to the data hungry nature of the deep learning frameworks. By taking the large pool of self-labelled data inherent in our self-supervised model the sheer amount of data needed for good results are satisfied, but by using this amount of data creates other problems. The time and resources it takes to pre-train the model increases substantially, necessitating the need for methods to shorten this process. The solution to this is then to pre-train the model or in other words divide the training process into two separate steps.

1. Pre-training: Distill general knowledge into the model about the input data with the use of a pre-training task.
2. Fine-tuning (Task-specific training): Task the model with predicting the specific outputs desired.

By keeping the general knowledge of the input data separated this shortens the training process, as these weights can be either imported or trained once and then reused when considering a different downstream task. This implementation saves both time and computational resources as the it shortens the fine-tuning training.

There are general models that are openly available like ResNet50 [15] for object detection or BERT [16], the original GPT [17] and its later iterations for Natural Language Processing (NLP) oriented tasks. These models works well but are modality specific. As such they are not adapted for input data from multiple domains such as different sensor modalities.

2.4.1 Pre-training Task

When conducting the pre-training of the model it has to be given a goal to train against. This pre-training task is meant to give the model an understanding of the features latent in the data that is provided to it. In the case of this thesis, the pre-training task is very much alike what the goal described in the report [6]. Tasking the model to predict the changes in the environment around the ego-vehicle from only its input data, the vehicle first has to understand what the changes in its surroundings representation mean. By tasking the model to understand its surroundings, the goal and intention is that the understanding of the input data is able to be reused for when the model is fine-tuned for later downstream tasks. The newly encoded information is represented by the loaded weights when the subsequent task-specific training starts. Therefore the information acquired during pre-training are used in

tasks that leverage this new inherent understanding.

2.5 Autonomous Driving

In this section AD and ADAS is presented as well as the importance of sensors and machine learning for vehicle perception.

2.5.1 AD/ADAS Systems

An example of a modular AD/ADAS architecture includes sensor hardware, data processing systems with focus on sensing as well as perception, a system for decision making and planning in addition to driving controls. The data processing systems are essential in order to achieve real-time understanding of the dynamic environment surrounding a vehicle, which is a requirement to realise the more advanced functionalities of AD/ADAS. This understanding then informs driving decisions and other automotive services available. ADAS functionalities can play a prominent role in increasing traffic safety with applications such as adaptive cruise control, parking assist, lane departure assistance and collision alerts. The fundamental techniques used in ADAS can be used as a base for fully realised AD systems [18].

2.5.2 Sensors in AD/ADAS Systems

Different types of sensors are used to form reliable sensing capability in ADAS, such as visual sensors like cameras, LiDAR, and RADAR. Cameras, which are usually more prevalent and cost-effective, are adept at providing essential information that allows further analysis. However, they fall short in their performance when it comes to being operated in different weather conditions. LiDAR are an active sensing method where a pulsed laser is utilised to measure distances based on the travel time of the return echo. While being much more capable in offering precise 3D sensing capability than image based sensors, they also come with higher cost and likely greater latency to make a scan of the environment. RADARs are similarly used to determine the position of surrounding objects but are also capable of measuring velocity in the radial direction. They are relatively cheap and can be less sensitive to different weather conditions. Each sensor modality has its advantages and disadvantages which could limit their utilisation when used individually without fusion. Therefore, one has to account for their heterogeneity when fusing different sensors [19][20].

2.5.3 Sensor Fusion

When constructing systems in charge of safety features, their reliability is of utmost importance. This reliability is as mentioned in the introduction, improved through the fusing of the multi-modal input data. The goal of sensor fusion methods is

to build comprehensive and robust sensor representations. This is implemented by combining output data from different sensors with complementing strengths and weaknesses in a way that optimizes the sensors individual characteristics. To fuse sensors, a crucial step is to calibrate sensors to have them sharing the same coordinate system. Additionally, consider that each sensor works differently and that their outputs are not all synchronized. Creating the problem that some data may be older than others and how that would describe the value of the input data. Once the raw sensor outputs are calibrated and aligned, one could consider fusing encoded sensor information. A fusion methodology must consider noises from all sensors throughout the process, since the inherent noise and errors from each sensor type is propagated through the fusion process [3].

2.5.4 Machine Learning in AD/ADAS

Computer vision and pattern recognition is a key part of ADAS, where determining the objects near a vehicle in 3D space is a required component of perceiving the surrounding environment. Practically, object detection is used to detect bounding boxes and categorizing objects of interest. DEtection TRansformer (DETR) [21], is a method based on viewing object detection as a set predicting problem used to streamline this process by utilising an encoder-decoder architecture based on Transformers as well as CNN. The model is trained end-to-end to predict objects in parallel with a set loss function using bipartite matching between ground-truth and estimated objects. This method minimizes the use of manually designed components for encoding prior information. The three main components are firstly a CNN backbone for extracting features and flattening the representation together with adding positional embeddings. Secondly an encoder-decoder Transformer where the decoder takes as input a small fixed amount of learned positional embeddings named as object queries which are transformed into an output embedding. Then finally the embedding is input to a Feed Forward Network (FFN) for the final prediction. A framework for 3D object detection using multi-camera is DETR3D [22], which is based on DETR. This method is implemented to calculate predictions directly in 3D space, unlike many other methods which instead estimates 3D bounding boxes from monocular images.

In order to distil information for multiple modalities such as for multi-modal sensor fusion, the Perceiver [8] model can be used. This is a model with utilises asymmetric attention mechanisms, which is also built on Transformers [4], in order to iteratively manage large amounts of inputs of different modalities. What separates this model is the focus on building an architecture which can handle high-dimensional inputs while retaining flexibility, as well as expressivity needed to deal with different modes of input data. Standard Transformers scale quadratically with the input which makes this model reduce complexity by decoupling the network depth from the input data through iterative attention of inputs. The Perceiver model essentially uses cross-attention over a supplementary low-dimensional matrix in order to achieve this where weights are optionally shared between iterations of the Transformer layers.

Furthermore, Unifying Geometric and Semantic Self-Supervised Pre-training for Autonomous Driving (GASP) is a self-supervised pre-training framework which builds on UnO and was developed to create a comprehensive understanding of the environment which surrounds a vehicle in a generalizable manner. This is done by predicting general occupancy, ego-occupancy as well as distilled high-level features from a vision foundation model. The model is used to reduce the dependency on human-labelled annotations in order to make use of the vast amount of unlabelled data which already exists in AD development. GASP learns a generalizable representation of the environment, which can in turn be used as input for downstream tasks. An advantage of this framework is the way it can integrate different sources of signals which are already available in vehicles [23].

2.6 Ego-motion as Supervisory Signal

This thesis is dependent on a supervisory signal that the machine learning model needs to help understand the changes from one point in time to another. This is what is fed to the model, the thing that trains it to relate motion with changes in its surroundings. The ego-motion is what describes the motion of the ego which for the case of this thesis is a car. While ego-motion entails both translation and rotation as represented in the matrix below (2.12). This is what is called an extrinsic matrix and is a term from computer vision that details the movement of external motion on the ”beholder” or ”camera” as it can usually likened to.

$$\begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}_{4\times4} \tag{2.12}$$

For the case of this thesis the part of the ego-motion and by extension the extrinsic matrix that will be the subject of scrutiny is the subset of the extrinsic matrix, the rotational matrix indicated by $R_{3\times3}$. The translation vector denoted by $T_{3\times1}$ is therefore not utilized for the intents and purposes of this thesis.

When the extrinsic matrix takes the shape of an identity matrix that indicates that no rotation has been done. In this thesis a comparison between two states of rotation is done and for that the matrices are given values that describe the movement relative to each other. The initial state is the identity matrix and the subsequent state describes the rotation of the ego-motion that happened between those two steps in time.

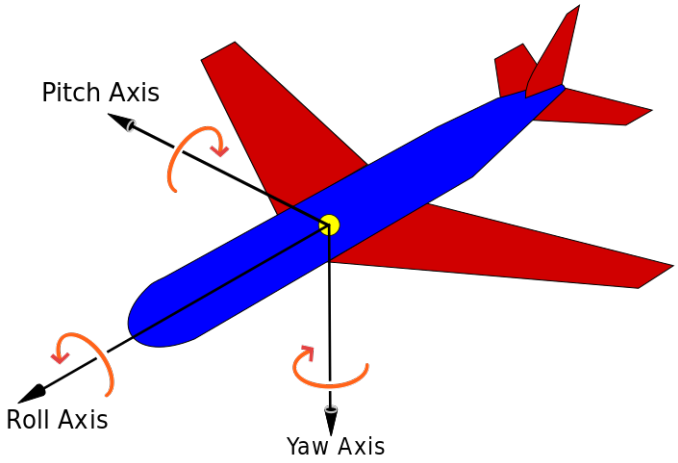


Figure 2.6: *Showcasing the rotations around each of the Cartesian coordinate system axes for ego-motion. Yaw is the rotation around the Z-axis, pitch about the Y-axis and roll around the X-axis.*

The rotation matrix can then be dissected and broken down into three separate matrices that each describe the rotation around one axis in the Cartesian coordinate system. These three axes are yaw, pitch and roll and can be seen represented in fig. 2.6 for a more intuitive explanation. Of these the yaw is the most important of the three, since the use case for this simulation is a car driving on the road. Yaw is the rotation around the Z-axis in the Cartesian coordinate system which would detail whether the ego-vehicle is doing a right or a left turn. But pitch, yaw and roll rotation can also be described with a 3x3 transformation matrix

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}_{yaw} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}_{pitch} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}_{roll} . \quad (2.13)$$

Where the rotation matrix $R_{3\times3}$, can be decomposed into rotations around their axle as seen in (2.13), whereby solving for the α, β and γ gives the angle in radians that each axis is rotated by. This along with the speed are fed to the model as supervisory signals.

2.6.1 Siamese Network with Ego-motion as Supervision

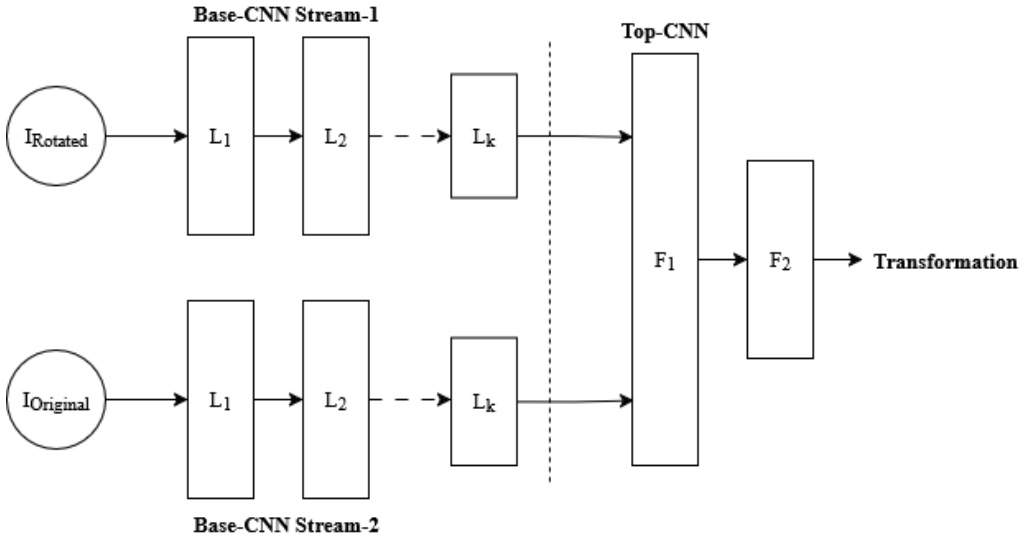


Figure 2.7: *Architecture of a Siamese network with ego-motion as supervision described in [6]. It takes two images as input, which differ only in orientation, and estimates the transformation between them using CNN for feature extraction. A supervision signal derived from ego-motion is used to guide the final prediction.*

The methods employed in this work build upon the approach proposed in [6]. In this study, a Siamese network architecture was used to incorporate self-supervisory signals derived from ego-motion, such as rotation and speed, to train the model. For visual tasks such as object detection and scene recognition, it was found that features learnt with this model were comparable to using features instead learnt using class-labels as supervision where there was limited label availability. This was evaluated using images and odometry data for a vehicle in urban traffic.

The network is presented with two slightly different views of the same scene as input, forming a pair of streams that share weights in a Siamese fashion. Each base stream processes its respective image through blocks of CNNs to create general feature representations. The streams differ only in the orientation of the input images, which are paired to reflect relative transformations between views.

Once the general feature representations are obtained, the outputs from both streams are compared and combined in a top layer where features used for final prediction is extracted, as illustrated in Figure 2.7. In this layer, the supervision signal of ego-motion is used, allowing the model to learn to predict the transformation between the two scene representations. This approach enables the network to learn meaningful visual features without relying on manually labelled data, demonstrating that ego-motion can serve as an effective supervisory signal for training vision-based models.

2.7 Performance Metrics

To quantify the performance of the downstream tasks we describe the model's predictions with a handful of metrics. The majority of the OD metrics are introduced and described by the research paper released alongside the NuScenes dataset [24]. The dataset has been popularised through the use of open competitions where companies, research labs and students can compete and submit their score. Therefore the need for a universal set of metrics to enable detailed comparisons between models. An overview of the metrics is presented in tab. 2.1. In this table lower values are preferable in all cases except for Average Precision (AP) where higher values are wanted.

- The most essential metric for OD is AP. AP stands for Average Precision which describes the precision of the object identification.
- Average Translation Error (ATE) - The Euclidean distance between the ground truth and prediction centre.
- Average Scale Error (ASE) - Aligns the bounding boxes and is calculated as 1 - IoU (Intersection over Union).
- Average Orientation Error (AOE) - The yaw angle difference between prediction and ground-truth in radians.
- Attribute Classification Error (ACE) - Absolute velocity difference in m/s.
- Identity Switch (IDS) - Indicates how often a detected object changes its unique identifier.
- Zero Detection Score (ZDS) - Scenarios where something should be detected but is not predicted, i.e missed.

As for the HT metrics they all describe the error between the best prediction path and the actual ground-truth path, Trajectory Displacement Error (TDE), at a certain point. All three HT metrics describe the same path but at different points. The error distance are evaluated at the first point, the 10th point and the 50th point, which would describe the accuracy from the initial start point down the line to the end point how accurate of an prediction it is.

Metric	Type	Measures
Object Detection		
AP	Detection	Precision-recall (detection)
ATE	Error	Position accuracy (m)
ASE	Error	Size estimation error
AOE	Error	Orientation angle error (radians)
ACE	Error	Attribute classification error
IDS	Tracking	Identity switches in tracking
ZDS	Tracking	Missed detections
Holistic Trajectory		
TDE at point 1	Error	Position accuracy (m)
TDE at point 10	Error	Position accuracy (m)
TDE at point 50	Error	Position accuracy (m)

Table 2.1: *A summarizing table of the relevant metrics for the downstream tasks of OD and HT and used to fine-tune the perception model.*

3
Methods

In this chapter, the methods applied to ensure the pre-training produced the best results in isolation are detailed. A representation is furthermore produced according to GASP with an output representing vehicle surroundings, which can be used as an input to the downstream tasks and to which the proposed pre-training is added. The performance of the pre-training model is then evaluated on fine-tuning a perception model [23].

The dataset used is similar to the ZOD which is a large multi-modal AD dataset collected over two years in Europe. The data was collected with a vehicle platforms with a full sensor suite including cameras, LiDARs and RADARs. The data includes frames, sequences and drives where a frame consists of 100k annotated samples, a sequence comprises 20 seconds sequence with full sensor coverage while a drive is instead defined as a 29 multi-minute sequence with full sensor coverage. The sensor setup of the ZOD dataset was identical on all collection vehicles which included a high-resolution camera, three LiDARs as well as a high precision GNSS/IMU. The data is also provides anonymization for all images in the dataset [7].

3.1 Pre-training Architecture

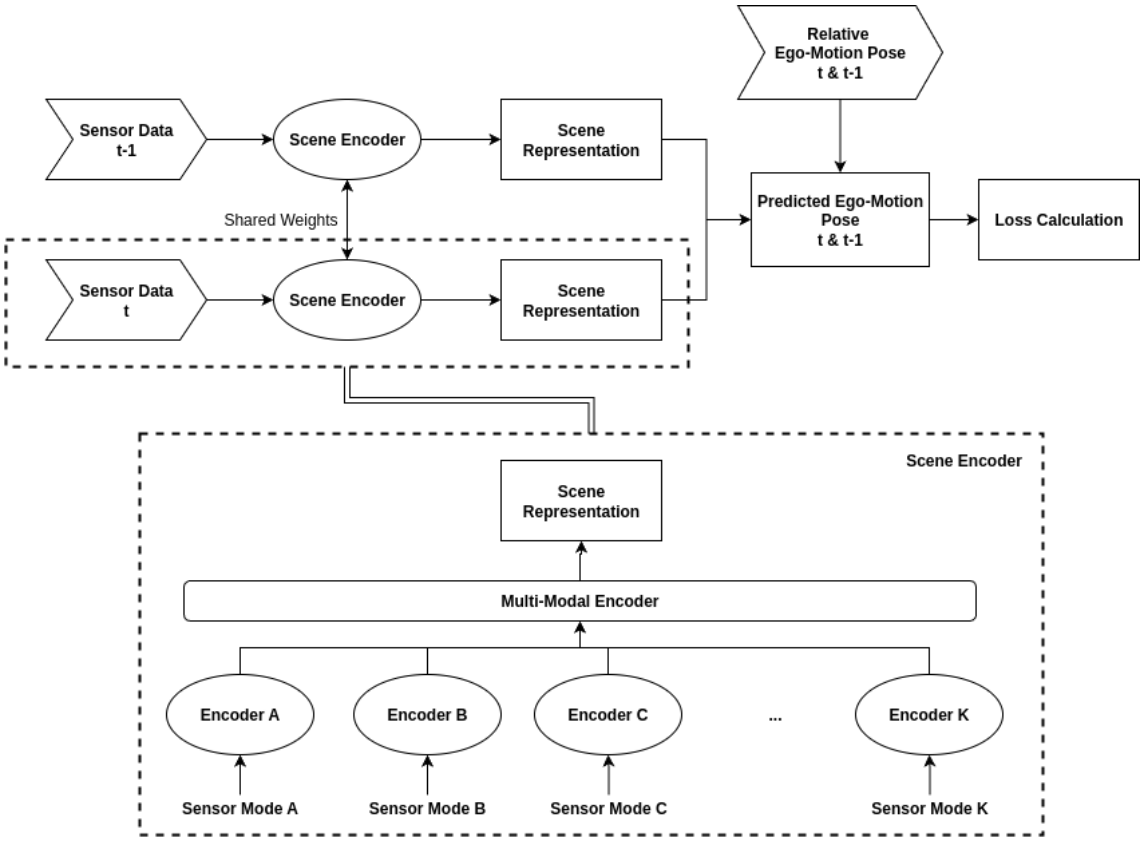


Figure 3.1: *The pre-training architecture, which includes two base streams encoding multi-modal input data from sensors into scene representations. This is then used to predict a relative ego-motion pose.*

The pre-training pipeline is presented in fig. 3.1, where each uni-modal encoder first encodes the individual sensor modalities. Then this is followed by the multi-modal encoder which fuses the information into a scene representation which can be modelled with high-dimensional vectors. This representation is what is used as input in the prediction of a relative ego-motion pose. In order to ensure that the relative pose is non-trivial where the transformation is an identity matrix, it must be sufficiently large temporal distance between the sensory inputs at t and $t-1$.

Essentially, the model has to learn how to fuse sensor information through uni-modal and multi-modal encoders. This means that the model has to learn how to reason about static scene elements while predicting the dynamic motion of the ego-vehicle. As such, the model is allowed to contextualize the scene understanding through differentiating between static and dynamic objects such as other moving vehicles. This should comprise high task relatedness for downstream tasks where static and dynamic objects should be separated.

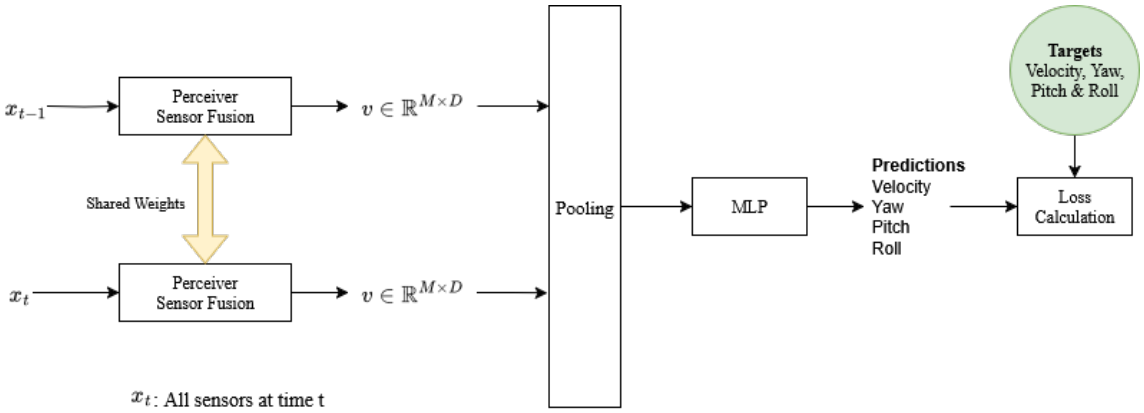


Figure 3.2: *The pre-training pipeline illustrates the fusion of inputs separated by temporal distance, from which the sensor fusion scene representation is pooled. This pooled result is then fed into a MLP head. The model uses ego-motion data as a supervision signal for the loss calculation.*

During training sensor data from the modalities of cameras, LiDAR and RADAR is used as input. The relative ego-motion pose is extracted by measuring the changing rate of yaw, pitch and roll as well as measuring the velocity longitudinally and latitudinally. These measurements are taken by sensors which has GNSS embedded and in all experiments the time between sensor poses is set to 1s. This is then represented with a transformation matrix which includes the rotation and the translation which in total has twelve elements of interest, see eq. 2.12. This element values in this matrix ranges between [-1, 1] in order to describe all possible motion, however, the values are typically close to 0 or ± 1 due to the identity matrix representing no change between two states. When the rotation matrix is an identity matrix the vehicle is moving along a straight road. In the case of the training in this thesis, the relative rotation is converted to relative angles of yaw, pitch and roll in radians while the translation is implicitly represented with vehicle velocity directly in m/s. The velocity is assumed to be 140 km/h or 40 m/s at its highest and normalized between 0 and 1. The expectation is that the characteristics of the relative poses depends on if a vehicle is driving slow or fast. Furthermore, it is expected that velocity and yaw have the highest impact on training.

The training objective is essentially a regression task where the model predictions of relative ego-motion poses will be regressed to the target values of interest. This means that any distance metric can be used to measure the loss. In this case the model measured the L_1 norm during loss calculation. The pre-training implementation can be seen in fig. 3.2 where the output from each base stream is aggregated through average pooling. This is then used as an input to a MLP with the input dimension of 256, hidden dimension of 512 and output dimension as either 4 or 2 together with a dropout rate of 0.2. During training an Adam optimizer was used.

3.2 Pre-training Design

Parts of the pre-training design was evaluated in isolation for curation of the dataset, configuration of target predictions as well as loss scaling. The best results from each were put together for the final design.

3.2.1 Data Filtering

Given that the machine learning model is dependent on ego-motion as its self supervision, providing the model with qualitative data is an utmost priority. By curating the data that the model aims to train for, we can help the model avoid overfitting by providing it with a diverse dataset. If the input data would consist mostly of for example drives on a highway, the simplest and most likely outcome would be for the model to assume that the next action would be to continue straight ahead.

So what is the basis for evaluating what is a desirable rotation and what is not? As was explained in the previous theory section, the rotation is described by an extrinsic matrix. The shape of the data in question is a series of extrinsic matrices where the first matrix in a set determines the starting point for the rest of the "sequence". The first matrix is an identity matrix and for every subsequent time-step the rotation between time-steps are expressed as a relative rotation. This means that if the rotation for a specific time-step is large enough it differentiates itself from a transition between matrices that experience no rotation relatively clear. Of course if the time-steps are to densely packed, erratic motion might get confused for being a part of a larger rotation, which is why we have chosen to take a more careful approach to the problem. As a way to safeguard that the measurements are correct, the interval between time-steps are currently set to one second between measurements. This can of course potentially miss some sudden outbursts of weaving or changing lanes in traffic but for the purpose of feeding the model with a varied set of rotations we deem it enough for the purpose of this paper and leave it to further scrutiny in the case that interest in this work persists. This is then through a built-in SciPy python package converted to radians describing the rotation from one moment in time to another. The desired quality of the following script is to curate the rotation described in radians so that its distribution has a flatter distribution and a smaller ”centre of gravity” towards the centre, i.e. zero/no rotation.

Algorithm Design

This algorithm was created with the intention to take the original distribution and flatten it. This would make it so that less common events such as turns and lane changes becomes more proportionally represented than the data that is most usually found when driving on a road, i.e going in a straight line. It has as input an extrinsic matrix, which as mentioned in the theory section contains the rotation matrix and a translation vector, at this time we are only interested in the rotation matrix.

An early attempt at implementing this design had evaluating the variance of a sequence over each rotation axis as the basis for distinguishing a good sequence from

a less desirable one. But as is probably known to the reader, variance only gives a description of the spread of values which was what was sought after. But there were cases where the sequence only contained a left or right turn, which did not always give a large enough spread of values. These sequences could not overcome the threshold value that was set to evaluate them to see what was discarded and which were kept to use as input to our model.

As a fix to the aforementioned problem the solution settled for was to take the absolute values of the rotation expressed as radians. This was then compiled both for each time-step to account for all the rotation around all axes, and over the sequence to account for the richness of data of the sequence as a whole. The summed up value for each sequence is then used to calculate a threshold value for which sequences are to be kept. When starting the algorithm a specific percentile is given that all chosen data has to surpass for it to be added to model input data. The algorithm then parses through all the calculated data and picks out all that is deemed enough/interesting. It does this by checking whether the sequences value surpasses that of the specified percentile.

Algorithm 1 The essential steps to the filtering algorithm

```
for 'Sequences' in 'Database' do
    Fetch needed data
    for 'Timestamps' in 'Sequence' do
        Convert extrinsic matrix to radians
    end for
    Sum over the absolute radian values in a time-step
    Sum over all timestamps to one total value
end for
for Sequence-index in list of sums do
    if list-of-sums(Sequence-index) ≥ threshold-percentile-value then
        add to new curated database
    end if
end for
```

3.2.2 Target Predictions

The pre-training was tested for two target prediction configurations. The first involved the four parameters of speed, yaw, pitch and roll while the second configurations only included speed and yaw, which were expected to have the highest impact on the training. Comparing these cases allows for evaluation of the impact pitch and roll have on performance and if they are worth the added model complexity.

3.2.3 Loss Scaling

The pre-training task for the machine learning model is to predict the speed and rotational angles between two streams of data. Given that specifically for the an-

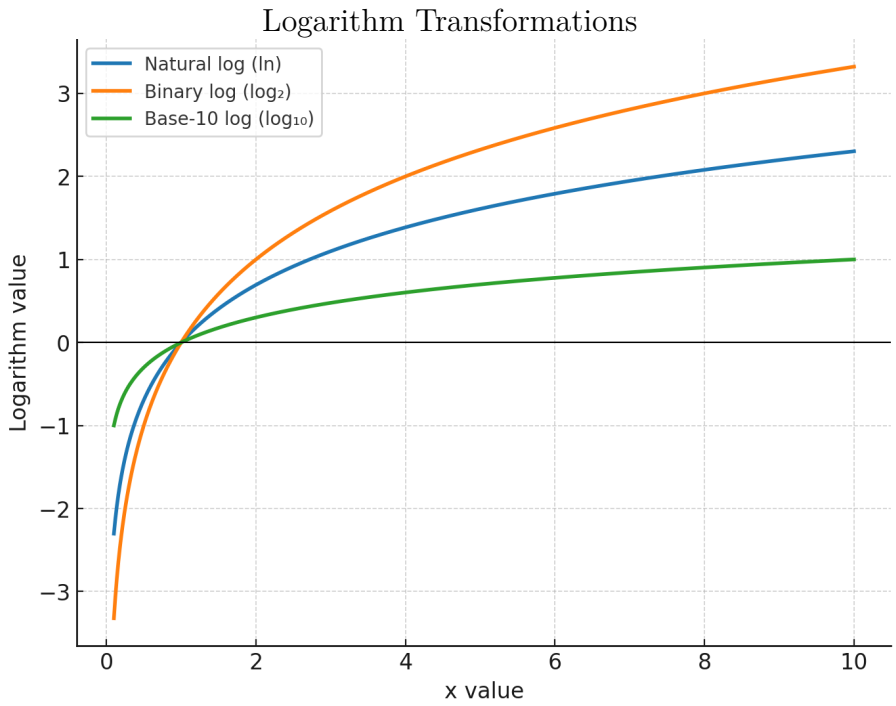


Figure 3.3: *The natural log function, note the steep incline as the x-axis is close to zero.*

gles, the magnitudes of the values range from π down to numbers several orders of magnitude smaller than one.

For this reason with the intent of helping to further separate angles on the smaller side, mathematical transformations have been tested out and applied to give the model better contrast to help differentiate similar data input.

The natural log transformation as is depicted in fig. 3.3 applied to the original Gaussian distribution of angles ”stretches” the distribution of small values further apart from each other and ”compresses” the distribution of larger values closer together. The resulting distribution from applying the natural log transformation to the angles Gaussian distribution helps create more separation to distinguish values whose magnitudes are on the smaller end.

The objective is to stretch the distribution of angles, so a steep incline for low values in the log function is the desired property. Out of the three most common log functions, two stand out, the binary log and the natural logarithm with base e. The functions both have a similarly steep slope, and therefore the natural log is chosen to be used. It is more common in the machine learning context and typically base 2 log is tied to binary numbers which is not present in our case.

Since the full range of possible angles is not particularly small to make sure the distribution becomes stretched and not compressed as they are for larger values, we need to normalize the numbers.

1. Shift the distribution for all values to be positive but not zero.
2. Normalize the range of values to be within some values to make sure the following transformation has the intended application.
3. Clip values on the lower end that possibly could be zero.
4. Apply the natural log transformation to the normalized strictly positive angles.

3.3 Fine-tuning Perception Model

The perception model which the weights of the pre-training was added to, was based on GASP. The downstream OD task used for evaluation is based on DETR and DETR3D, which are described in sec. 2.5.4. During these tests, a baseline was first established, which utilised weights trained on annotated data. This was then compared with a perception model where the weights were exchanged with those using the annotation-free pre-training weights. A good outcome would be if the model using annotation-free pre-training was able to achieve a similar or better performance. This would still be the case should the annotation-free model require a lot more data than in an annotated pre-training, as there exists a lot more unannotated data which can be made use of immediately by vehicle developers. As the training target is ego-motion based on odometry data, vehicle sensor datasets also have odometry data inherently available.

The perception model is fine-tuned on the downstream tasks of OD and HT. The methods used for the case of annotation-free pre-training was designed based on the filtering, target configuration and loss scaling parameters which had the highest performance in each isolated evaluation.

4

Results

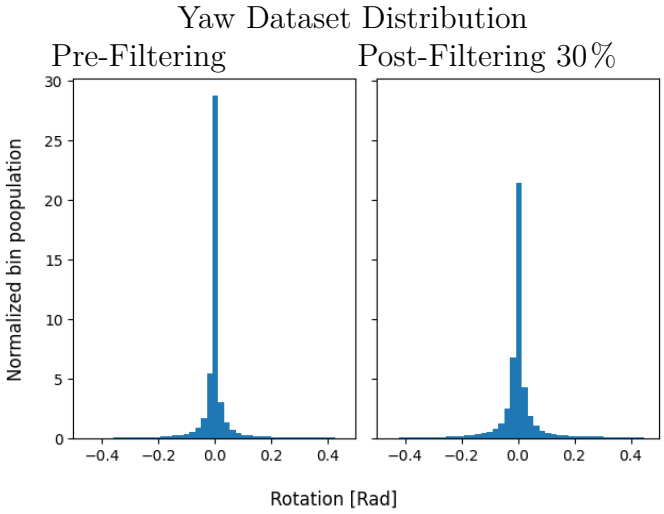
An important thing to note about the graphs and values in the coming result sections, is that the original values have been normalized based on the largest value present in the graph of the metric. These changes were applied as an anonymisation pre-caution as to not reveal confidential information. Nevertheless the information to take away from the following graphs and tables should be what works and the impact of different tweaks to the machine learning model.

4.1 Pre-training Design Evaluation

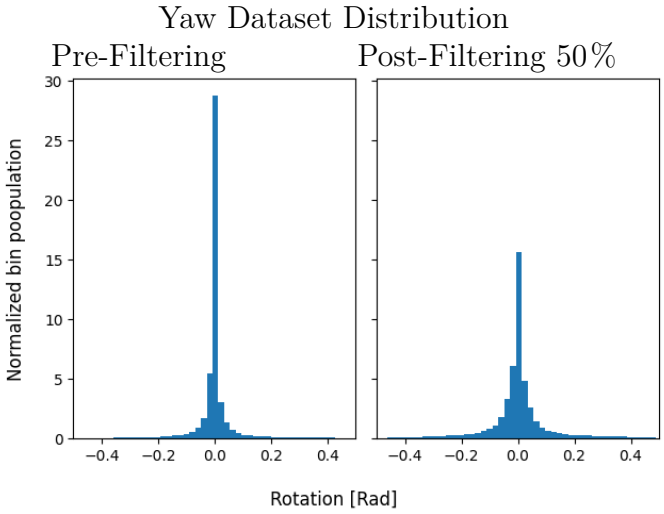
The results of evaluating the pre-training design for curation of the dataset, different configurations of target predictions as well as different loss scalings.

4.1.1 Result of Dataset Filtering

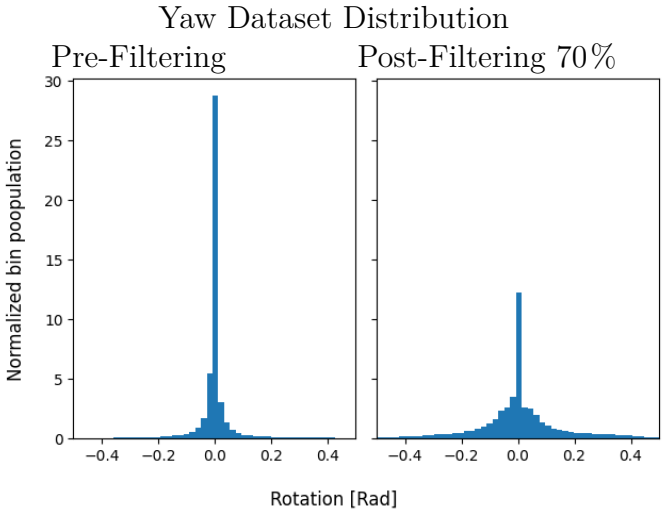
In the images from fig. 4.1, the filtering algorithm that was described in Section 3.1 is used to filter the ego-motion input. The three different scenarios investigated are those where 30 %, 50 % and 70 % of the data points with the least amount of absolute rotation are dismissed. All the figures in this section are normalized, i.e., only the relative values are shown, to anonymize the data. Therefore, the connection to actual values in radians are gone, but the characteristics of the distribution are still present.



(a) *Filtering has removed 30% of the data closest to zero.*



(b) *Filtering has removed 50% of the data closest to zero.*



(c) *Filtering has removed 70% of the data closest to zero.*

Figure 4.1: *The distribution of the absolute sum of rotation over all rotational axes pre- and post filtering for the largest 70% of rotations. Each subfigure corresponds to a different filtering threshold.*

Here we see how the evaluation criteria are applied to the absolute sum of all axes rotation. All values below the criteria set out by the value of the specified percentile is dismissed.

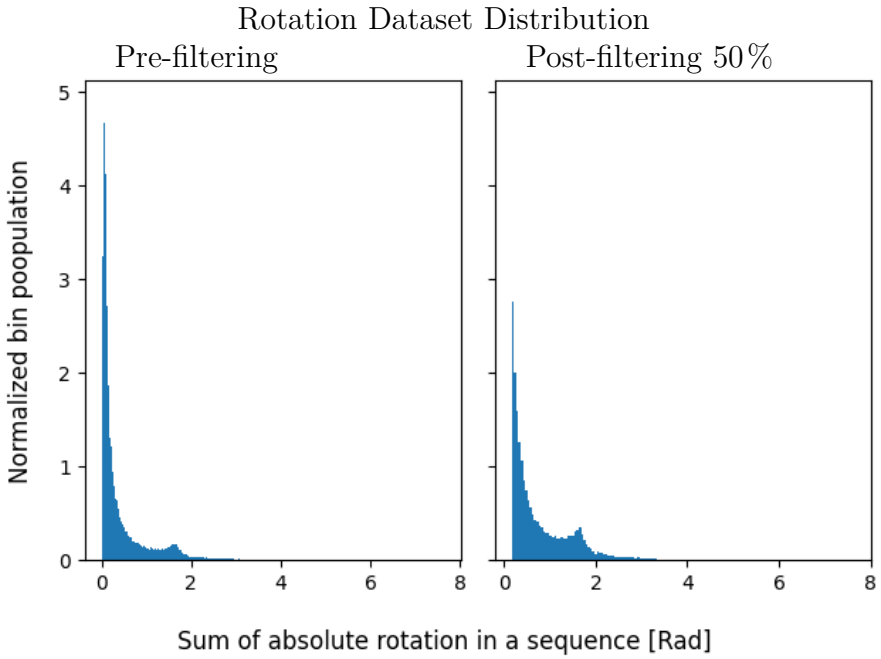


Figure 4.2: *This plot showcases the raw rotation distribution of each axis original rotation expressed in radians. Same filtering and data as the 50% scenario in fig. 4.1.*

Other information in the dataset not pertaining to ego-motion has also been inspected before and after the filtering. This had to be done to ensure that the filtering on one data type did not unintentionally exclude or distort the distributions of other data used for other downstream tasks.

4.1.2 Result of Dataset Filtering

Figure 4.3: *A comparison between different data filtering thresholds when trying to predict the yaw rotation. The models were trained for 20 epochs. Since the training takes a sizeable time to complete, the results are an example of a typical run, and outliers can occur.*

The training losses in fig. 4.3 follows for the most part intuition and the intention behind that implementation in the first place. The training loss has the smallest magnitude when the dataset is the least curated. On the other end of the filtering spectrum, the model has the largest training loss when its the most harshly curated. This happens since the uncurated datasets feeds the model with data that closely resemble each other. These are, for the sake of predicting rotation, the more trivial cases as they contain little to no rotation. While the more curated datasets consists of more data where twists and turns are more prevalent. Therefore, creating cases that provides more rotational diversity contributes more to a better understanding for the model.

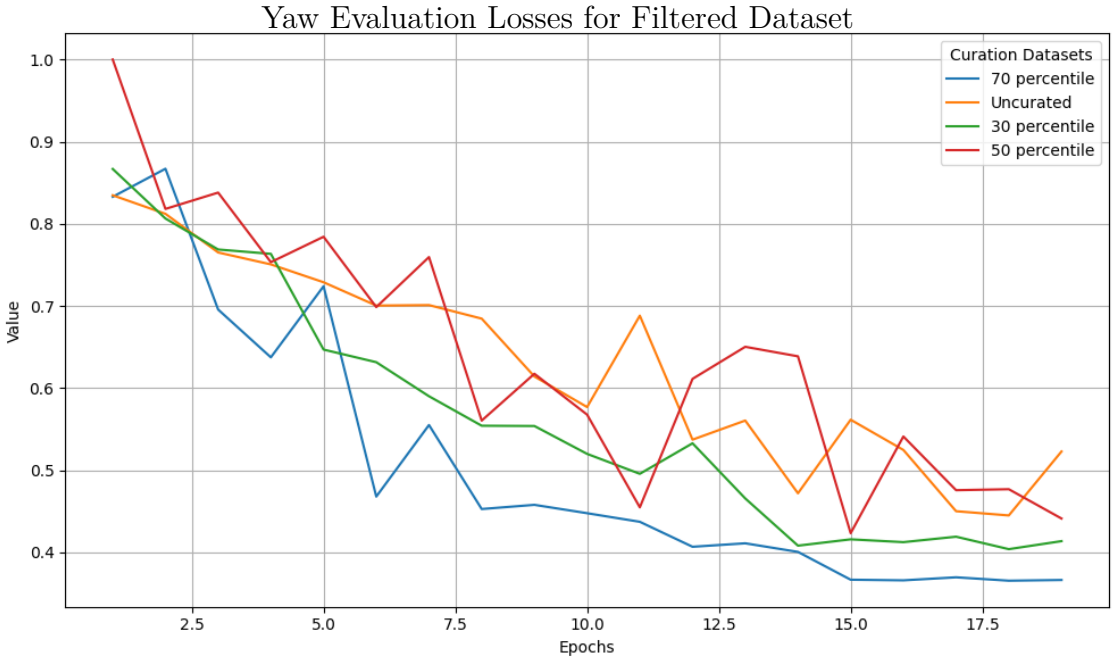


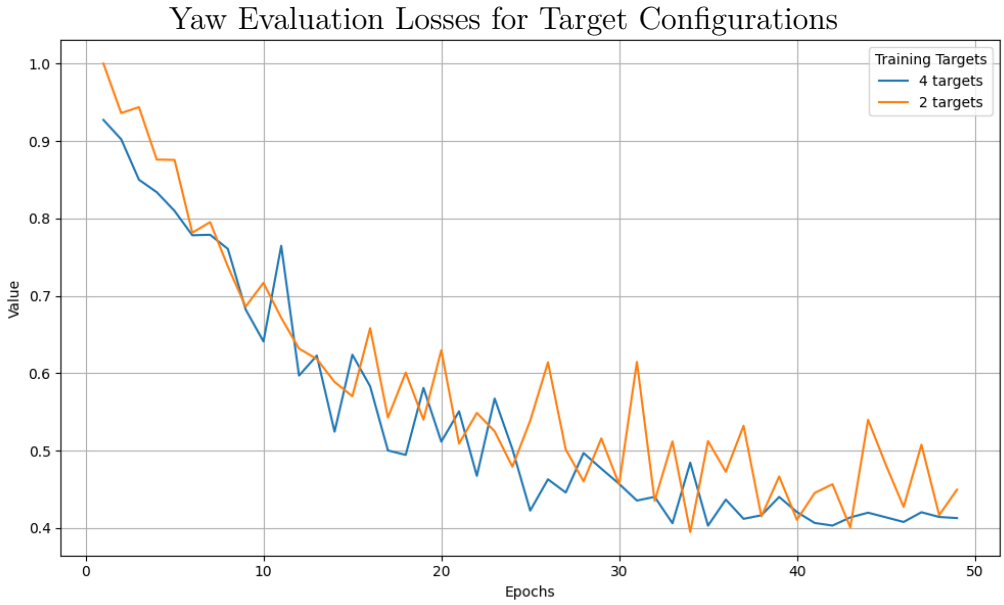
Figure 4.4: *A comparison between different filtered datasets on the evaluation of the yaw rotation. The models were trained for 20 epochs. Since the training takes a sizeable time to complete, the results are an example of a typical run, and outliers can occur.*

The models that were trained upon more diverse datasets is also able to handle outlier targets better, and therefore has a lower evaluation loss. Now, a reader with a keen eye can see that the evaluation loss plot in fig. 4.4, its not a guarantee but regard the plot more as an indicator of trends. But the explanation provided for that is the volatility of the smaller amount of data in the evaluation set, and the shorter run time of the provided graph in contrast to most other graphs in this report.

4.1.3 Result of Target Configurations

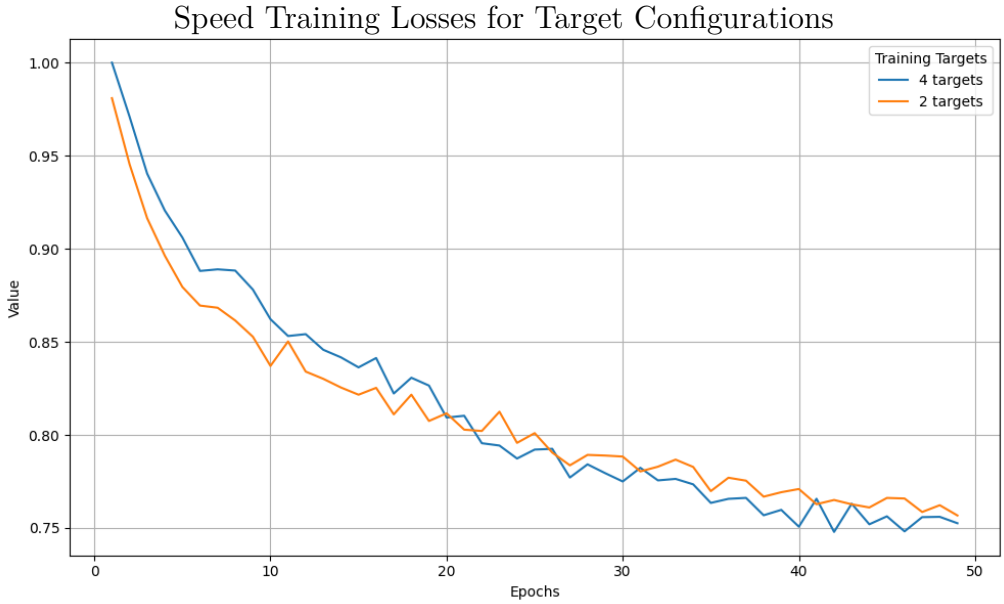
The model can be modified as to what and how many inputs that are given. In the coming section the impact of two different setups of input is evaluated. The simplest of the two takes just the input of the speed and the yaw rotation as these are what we considered the most integral parts to understanding the movements of the ego-vehicle, i.e a car in our use case. The opposing scenario contains the same targets that we are training against as before but all rotations around the 3-D axes are included. By including the pitch and roll in the other iteration of our model, it makes a total of four training targets for our model. The following figures fig. 4.5 and 4.6 sets the two models up against each other. This is to see which one performs the best in the task of predicting the targets present in both configurations, the yaw and speed targets.

(a) *Yaw training losses for different target configurations.*

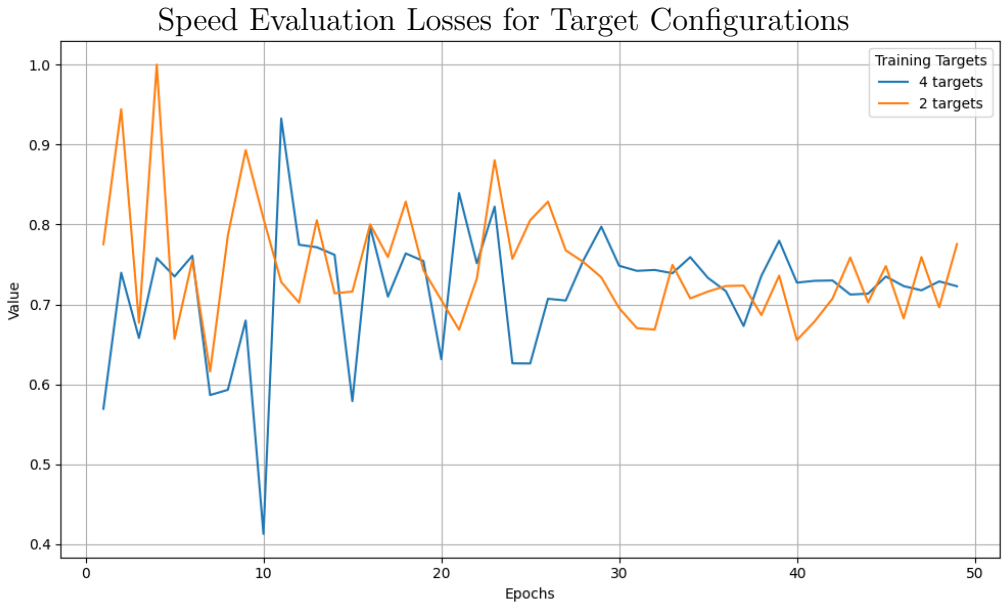


(b) *Yaw evaluation losses for different target configurations.*

Figure 4.5: *The training and evaluation losses for the different target configurations of the model and how it impacts the yaw training.*



(a) *Speed training losses for different target configurations.*



(b) *Speed evaluation losses for different target configurations.*

Figure 4.6: *The training and evaluation losses for the different target configurations of the model and how it impacts the speed training.*

In fig. 4.5 and 4.6 we can see produces comparatively equal results for the training loss. In the evaluation loss a generally slightly smaller loss is observed and also a loss curve whose variation decreases over time.

Therefore while the performance of the two targets is comparable to the four targets, the more information that is provided to the model, as ones intuition might suggest, takes the edge in performance. But as both cases are proved to be viable there could be some use case where a reduced amount of targets would be a preferred solution.

For the remainder of results in this reports all four targets are taken into account for our pre-training tasks.

4.1.4 Result of Loss Scaling

There are three methods that will be evaluated in this section:

- Original
- Unit normalized
- Iterated normalization

All models are pre-trained but that is where the similarities end. In the original model the angles have not been tampered with and is used mainly to evaluate. Unit normalized has the distribution shifted by π to move the theoretical range of the angles above zero, and normalized between zero and one. However the actual range of values is not that of the theoretical range, as it is a fair bit smaller making the distribution still heavily grouped close to the middle. The iterated normalization then takes it one iteration further, the distribution is once again shifted so that the all angles are positive but just barely. How far the distribution was to be shifted was determined through dataset analysis of it's pre and post filtration histograms from fig. 4.1b. Like the previous method the range is then normalized between zero and one. Since the iterated method had values spread throughout its' whole range, the hope was that the log transform would apply more stretch to their distribution.

Metric	Original	Unit normalized	Iterated normalization
Object Detection			
AP	-15.8	**100**	-15.3
ATE	+10.0	**100**	+9.64
ASE	+8.0	**100**	+1.6
AOE	+3.2	**100**	+1.2
ACE	+25.2	**100**	+11.3
IDS	+4.0	**100**	+207.3
ZDS	**100**	+10.7	+1.1
Holistic Trajectory			
TDE at point 1	**100**	+10.8	+9.1
TDE at point 10	**100**	+10.0	+12.9
TDE at point 50	**100**	+10.2	+6.9

Table 4.1: *Values in the table represent the final relative difference compared to the best value in the last epoch as percentages. The values across a metric are relative to* ***100*** *which denotes the best performing model. The different models were all trained with the 50% curated dataset. Bold numbers indicate the best performance while other values indicate relative change, metric definitions displayed in tab. 2.1. Plots of the training are attached in Appendix A.*

As can be noted by the numbers in bold font in tab. 4.1, the method providing the

best performance varies across which types of metric is focused on. The iterated normalization method was intended to lead to a performance increase upon the unit normalized method, but the results tells that it failed to improve upon its previous iteration over all metrics. Since the iterated normalization failed to present the best result across all evaluated metrics, the remaining two methods are the focus of the performance evaluation. The unit normalized method provides the better performance of the two in the OD metrics except for the ZDS metric. While the pre-trained model using the original input data provides the best performing result over the ZDS and HT metrics.

4.2 Performance of Fine-tuning Perception Model

In this section, the final model results are presented and compared against a model baseline. The model in its final state consists of:

- Targets: 4 targets
- Dataset: 50% curated dataset
- Radian transformation: Unit normalization

These are the settings for our model that through the previous result sections we have proved provide good results. The slight difference is that the 50% and not the 70% dataset is used. The reason for this is that the model requires large amounts of data for the pre-training. To that end a compromise is done and a larger dataset is retained while the data diversity is diminished slightly.

Given the amount of time and resources needed to carry out the different iterations of our model constrained the amount of runs that could be attempted. Therefore multiple runs of the same setup was not carried out. If a larger amount of runs would have been made, a larger amount of confidence could be bestowed upon the results to not include an outlier result. The only exception to this was the model baselines which can be seen in the figures below. Two different runs with varying performances were used to create an average of both to have a more solid foundation to compare against.

4.2.1 Performance of Object Detection

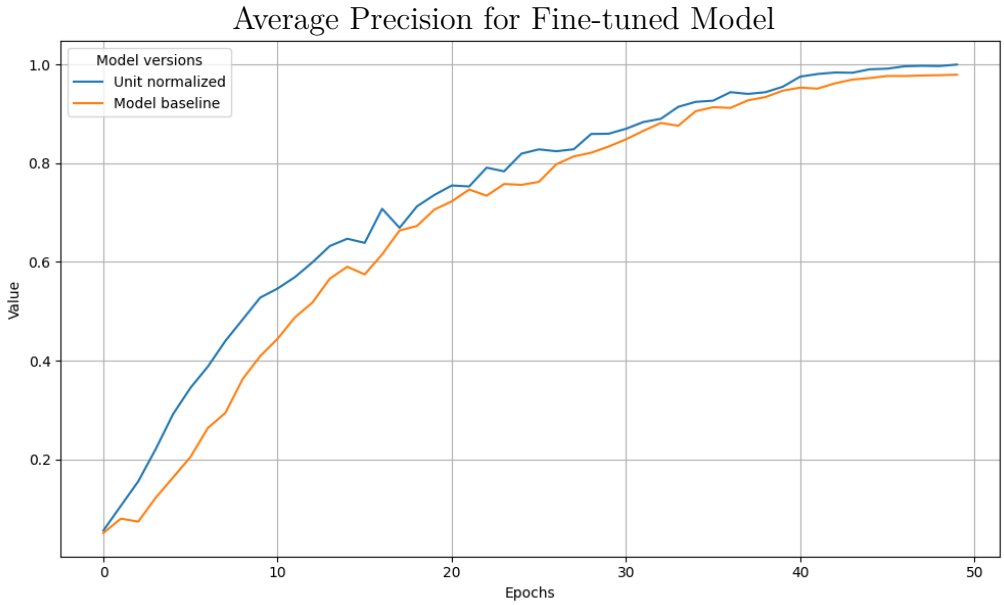


Figure 4.7: *Comparison between the model baseline and the pre-trained model on relative AP.*

The pre-training model was evaluated by replacing the weights for a perception model. A baseline for the perception model is also presented. The perception model was evaluated for the metrics AP, ATE, ASE, AOE, ACE, IDS and ZDS.

As presented in fig. 4.7, the AP of the fine-tuned models were comparable to each other and had similar increasing characteristics throughout the training. The model pre-trained on annotation-free data had slightly higher values, which indicated a slightly higher performance.

As shown in fig. 4.8, the ATE of the fine-tuned models were comparable to each other and had similar decreasing characteristics throughout the training. The model pre-trained on annotation-free data had a lower value throughout training, which indicated a slightly higher performance.

Further described in fig. 4.9, the ASE of the fine-tuned models were comparable to each other and had similar decreasing characteristics throughout the training. The model pre-trained on annotation-free data had slightly lower values, which indicated a slightly higher performance.

As displayed in fig. 4.10, the AOE of the fine-tuned models were comparable to each other and had similar decreasing characteristics throughout the first half of training. The model pre-trained on annotation-free data converged at a higher value, which indicated a worse performance.

In fig. 4.11, the ACE of the fine-tuned models. as presented, were comparable to each other and had similar decreasing characteristics throughout the training. The model pre-trained on annotation-free data converged on slightly higher values, which indicated a slightly lower performance.

As presented in fig. 4.12, the IDS of the fine-tuned models were comparable to each other and had similar increasing characteristics throughout the training and converged on similar values. The increasing values indicated an increasingly worsening performance.

Finally, as depicted in fig. 4.13, the ZDS of the fine-tuned models were comparable to each other and had similar increasing characteristics throughout the training and converged on similar values. They both had a steep lowering in value in the early stages of training. The stable but increasing values which indicated an increasingly worsening performance.

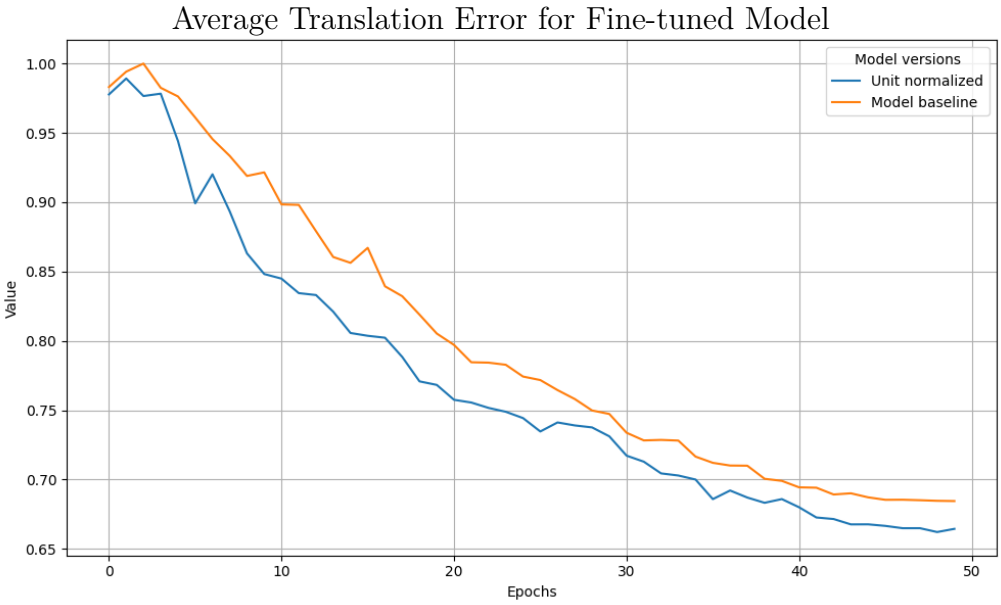


Figure 4.8: *Relative ATE for the fine-tuned baseline and proposed pre-trained.*

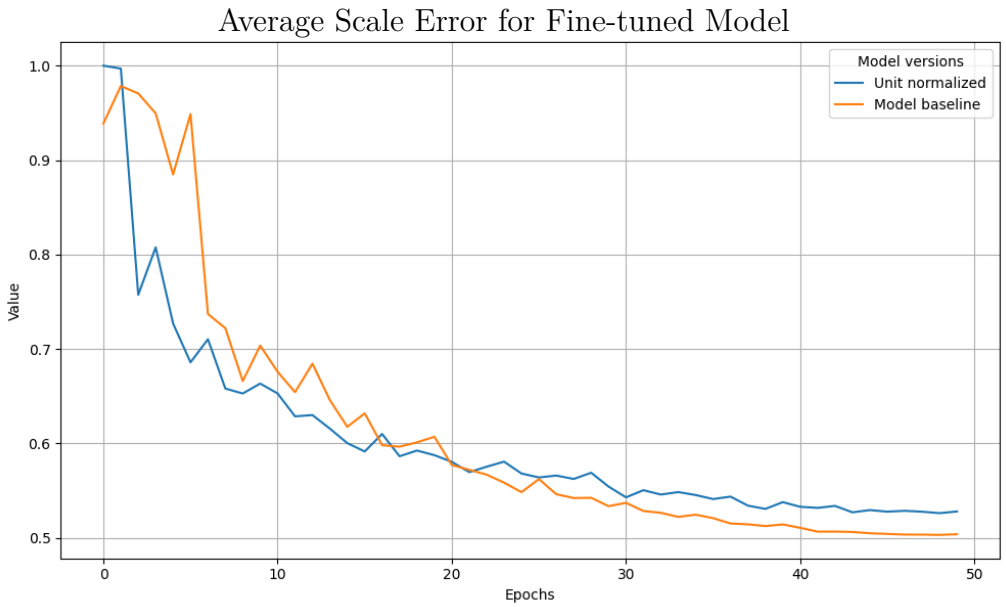


Figure 4.9: *Relative ASE for the fine-tuned baseline and proposed pre-trained.*

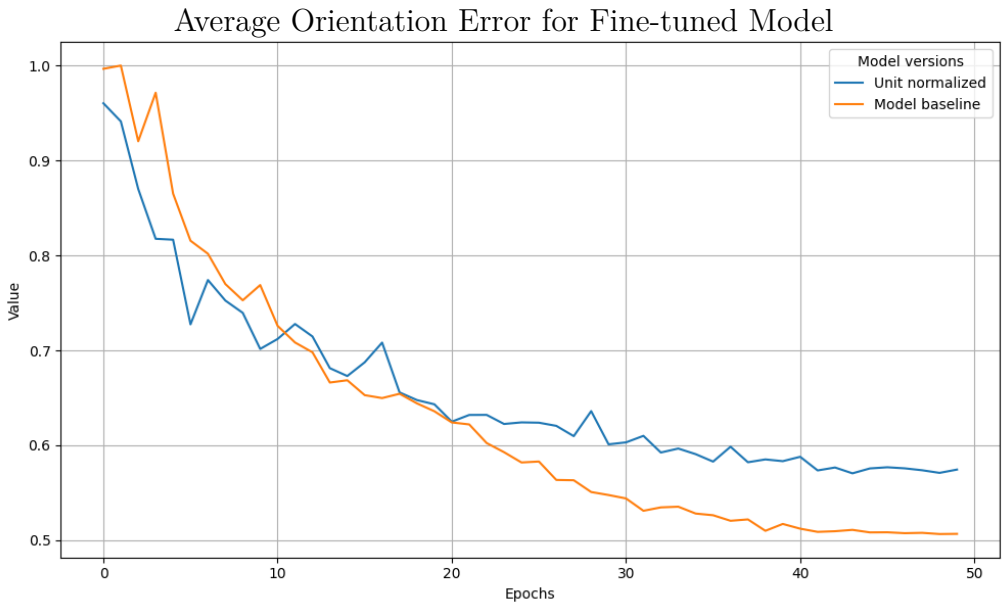


Figure 4.10: *Relative AOE for the fine-tuned baseline and proposed pre-trained.*

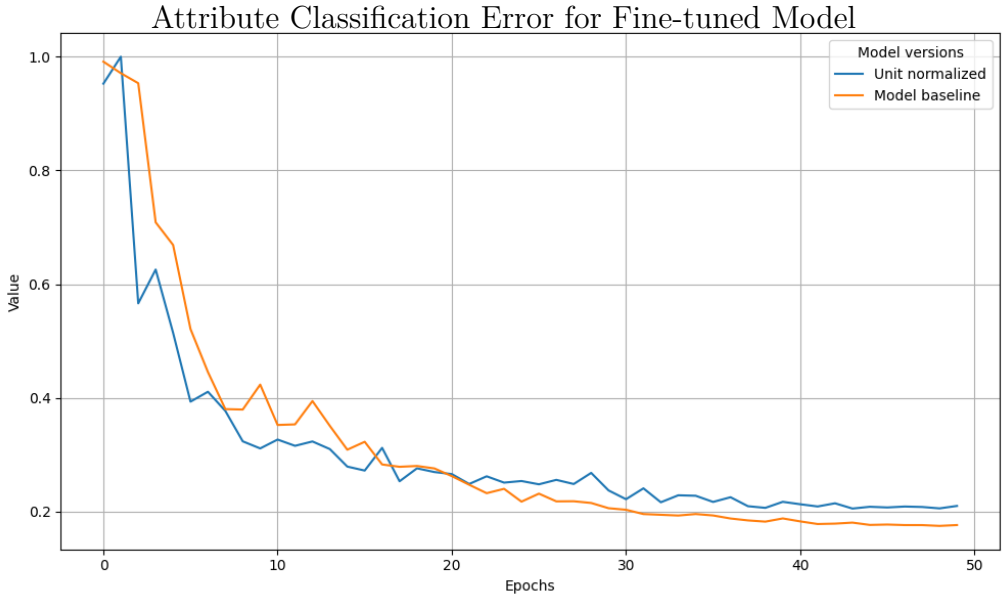


Figure 4.11: *Relative ACE for the fine-tuned baseline and proposed pre-trained.*

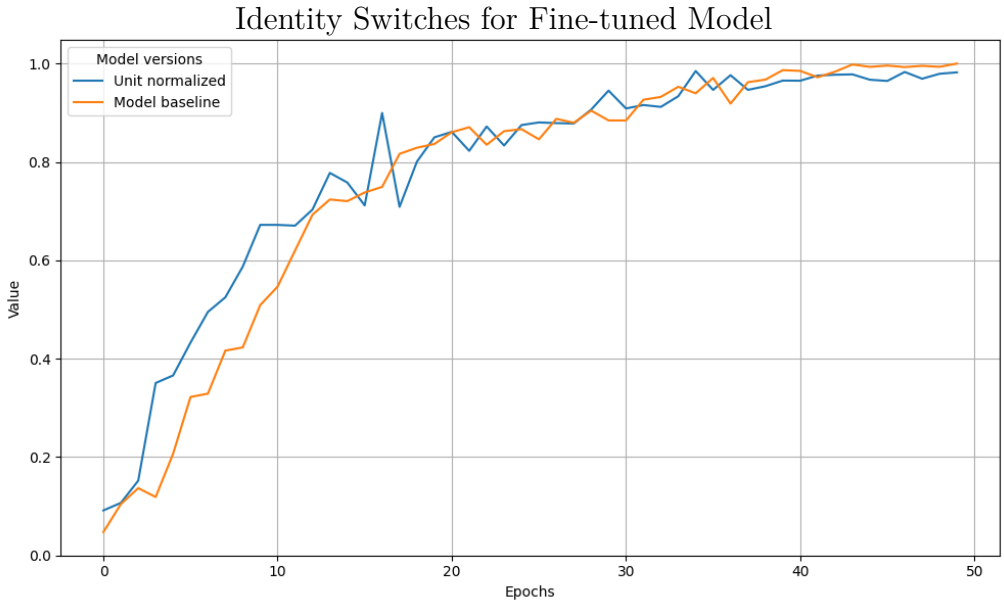


Figure 4.12: *Relative IDS for the fine-tuned baseline and proposed pre-trained.*

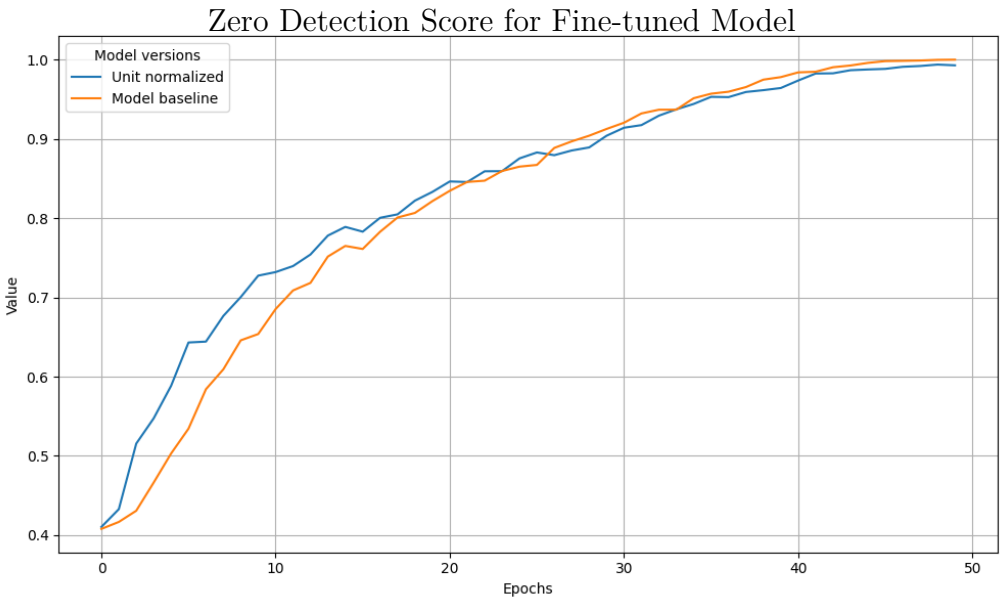


Figure 4.13: *Relative ZDS for the fine-tuned baseline and proposed pre-trained.*

4.2.2 Performance of Holistic Trajectory

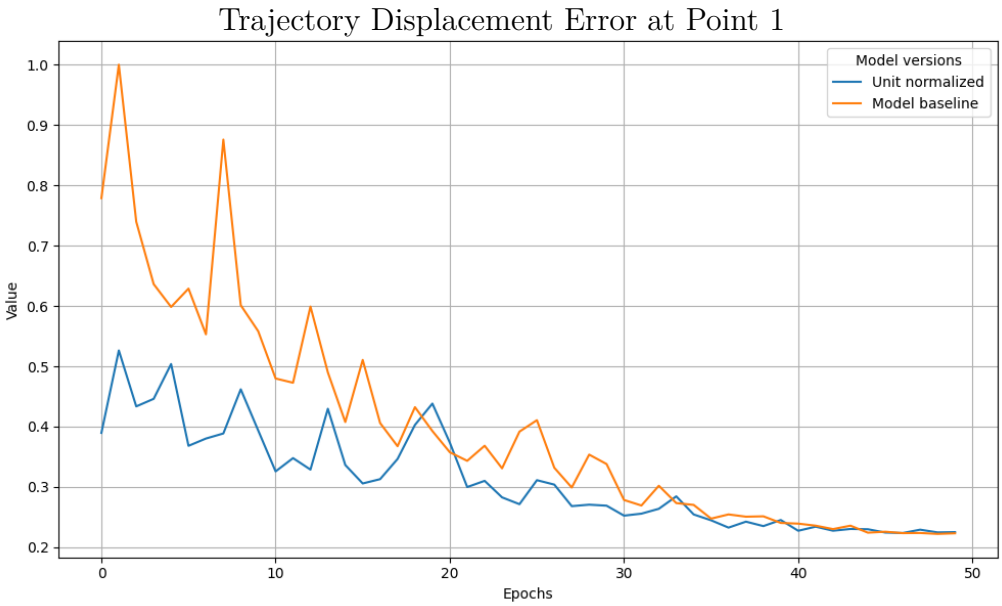


Figure 4.14: *Comparison between the model baseline and the pre-trained model on relative TDE at point 1.*

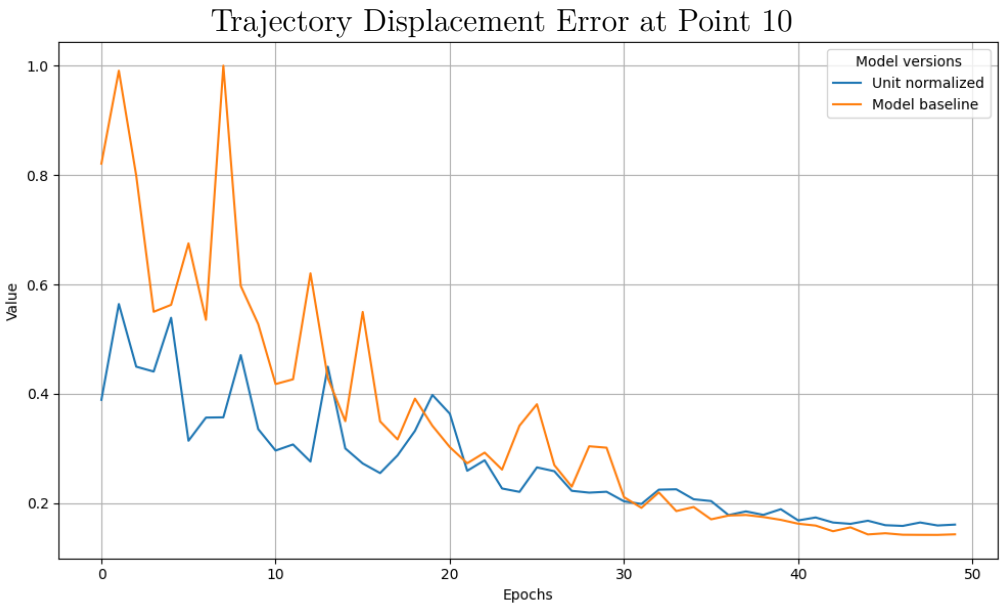


Figure 4.15: *Comparison between the model baseline and the pre-trained model on relative TDE at point 10.*

The prediction of the HT was also evaluated with TDE at points 1, 10 and 50. Graphs depicting the relative error for the perception baseline as well as the annotation-free pre-trained version is shown in fig. 4.14, 4.15 and 4.16. The model using the annotation-free pre-training is shown to have a large advantage of around 40% for all three points. The advantage continues for twenty epochs, however, in the end both models converge at similar values.

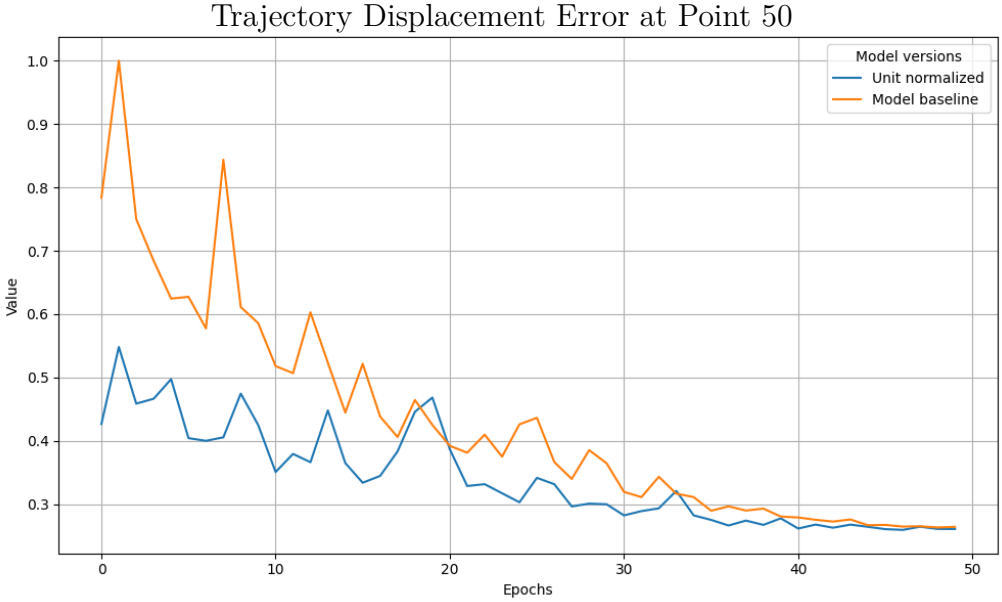


Figure 4.16: *Comparison between the model baseline and the pre-trained model on relative TDE error at point 50.*

4.2.3 Performance Summary

In general, the perception model using annotation-free weights performed slightly better than the baseline they were similar in most cases. This was in particularly true in the early stages of training for some metrics. This was, however, not the case for AOE where the baseline converged at a lower value. Though most metrics trended in the wanted direction, the metrics of IDS and ZDS had an unwanted increasing behaviour. Furthermore, ASE and ACE had sharp decrease in value in early stages in training for both model versions.

5
Conclusion

The results presented in Chapter 4 indicates that the proposed self-supervised pre-training approach can achieve performance compared to a supervised pre-training baseline, without using any labelled data. This demonstrates the viability of the approach in its simplest iteration and motivates further exploration to improve its capabilities.

A primary area for improvement lies in the method of data aggregation during pre-training. Currently, temporal frames are fused using mean pooling, which may lead to the loss of critical scene-specific information. Replacing mean pooling with cross-attention between consecutive frames could help the model capture more details for feature learning. This is important as the original method using a single modality had a known difficulty in distinguishing between static and dynamic objects and more information could be required to offset this weakness. This attention mechanism could also be of use in evaluating the model, as it could provide insights into what features and their related positional embedding the model attends to and thus places importance on. Another important direction is the systematic filtering and selection of training data. Investigating which type of ego-motion scenarios are under-represented in the dataset would improve the model's ability to generalise across diverse driving scenarios and environments. Increasing the number of training runs would also enable more statistically significant conclusions, providing a stronger basis for evaluating the pre-training method's effectiveness. Furthermore, examining the impact of sample size on pre-training performance could determine whether the model can achieve robust feature learning even with larger or more complex datasets, while taking advantage of the inherently annotation-free nature of vehicle ego-motion data.

Another area is the evaluation of different time intervals between input frames during pre-training. In this work, a one second interval was chosen heuristically, but systematically testing especially shorter intervals, could provide insights into how temporal resolution affects feature learning and the model's ability to capture dynamic scene information. Especially given that sensor samples, defining the possible limits in how short an interval can reasonably be, are provided with much smaller intervals. This would also need to be evaluated with the needs of the downstream task in mind. Furthermore, a shorter time interval would mean even smaller numbers when describing the relative ego-motion.
Future work could also explore pre-training directly on elements from the extrinsic matrix describing ego-motion instead of using the derived parameters such as speed,

yaw, pitch and roll. This may provide the model with richer supervisory signals and improve its correlation with downstream tasks, which could make the added training time worth the trade-off. Model complexity and architecture are further areas for investigation. Increasing hidden dimensions in the MLP head, experimenting with other head design types as well as exploring different loss functions could improve the model and its ability to capture fine-grained information. It is also of importance to determine whether the model relies on a single modality or if it manages to effectively integrate all available sensor inputs. Finally, it could be possible to extent the architecture to compare more than two temporal frames, for example, by introducing more base streams and then cross-attend features related to multiple time-steps in the data aggregation process.

Together, these suggestions provide a possible roadmap for advancing the pre-training methodology while maintaining its core advantage of leveraging large amounts of annotation-free data to produce multi-modal representations suitable for downstream ADAS tasks.

Bibliography

[1] WHO. "Road traffic injuries," Accessed: Dec. 20, 2024. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[2] European Climate, Infrastructure and Environment Executive Agency (European Commission), *EU road safety: towards "Vision Zero"*. Publications Office of the European Union, 2022, ISBN: 978-92-9208-143-0. Accessed: Dec. 20, 2024. [Online]. Available: https://data.europa.eu/doi/10.2840/701809.

[3] N. M. Deepika, B. Rajalakshmi, G. Nijhawan, A. Rana, D. K. Yadav, and K. A. Jabbar, "Signal processing for advanced driver assistance systems in autonomous vehicles," in *2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, ISSN: 2687-7767, vol. 10, Dec. 2023, pp. 1521–1526. DOI: 10.1109/UPCON59197.2023.10434599.

[4] A. Vaswani et al., *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762. Accessed: Feb. 27, 2025. [Online]. Available: https://arxiv.org/abs/1706.03762.

[5] B. Agro, Q. Sykora, S. Casas, T. Gilles, and R. Urtasun, *UnO: Unsupervised occupancy fields for perception and forecasting*, Jun. 12, 2024. DOI: 10.48550/arXiv.2406.08691. Accessed: Aug. 13, 2025. [Online]. Available: http://arxiv.org/abs/2406.08691.

[6] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago: IEEE, Dec. 2015, pp. 37–45, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.13. Accessed: Feb. 3, 2025. [Online]. Available: http://ieeexplore.ieee.org/document/7410370/.

[7] M. Alibeigi et al., *Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving*, Oct. 21, 2023. DOI: 10.48550/arXiv.2305.02008.

[8] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, *Perceiver: General perception with iterative attention*, Jun. 23, 2021. DOI: 10.48550/arXiv.2103.03206. Accessed: Feb. 18, 2025. [Online]. Available: http://arxiv.org/abs/2103.03206.

[9] *Improving global road safety, a/RES/74/299*, Aug. 31, 2020. Accessed: Feb. 21, 2025. [Online]. Available: https://docs.un.org/en/A/RES/74/299.

[10] I. Goodfellow, A. Courville, and Y. Bengio, *Deep learning* (Adaptive computation and machine learning). Cambridge, Massachusetts: The MIT Press, 2016, 1 p., ISBN: 978-0-262-33737-3.

[11] O. I. Abiodun et al., "Comprehensive review of artificial neural network applications to pattern recognition," *IEEE Access*, vol. 7, pp. 158 820–158 846, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2945545. Accessed: Mar. 3, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8859190.

[12] T. Nithya, V. Nivas Kumar, G. S, S. Deepa, V. C. M, and R. Siva Subramanian, "A comprehensive survey of machine learning: Advancements, applications, and challenges," in *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, Trichy, India: IEEE, Aug. 23, 2023, pp. 354–361, ISBN: 9798350325799. DOI: 10.1109/ICAISS58487.2023.10250547. Accessed: Feb. 27, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10250547/.

[13] D. Chicco, "Siamese neural networks: An overview," in *Artificial Neural Networks*, H. Cartwright, Ed., vol. 2190, New York, NY: Springer US, 2021, pp. 73–94, ISBN: 9781071608258 9781071608265. DOI: 10.1007/978-1-0716-0826-5_3. Accessed: Aug. 12, 2025. [Online]. Available: https://link.springer.com/10.1007/978-1-0716-0826-5_3.

[14] X. Han et al., "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021, ISSN: 26666510. DOI: 10.1016/j.aiopen.2021.08.002. Accessed: Aug. 29, 2025. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666651021000231.

[15] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. Accessed: Sep. 4, 2025. [Online]. Available: https://arxiv.org/abs/1512.03385.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. Accessed: Sep. 4, 2025. [Online]. Available: https://aclanthology.org/N19-1423/.

[17] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., "Improving language understanding by generative pre-training," 2018.

[18] A. Pandharipande et al., "Sensing and machine learning for automotive perception: A review," *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11 097–11 115, Jun. 2023, ISSN: 1558-1748. DOI: 10.1109/JSEN.2023.3262134. Accessed: Feb. 21, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10089400.

[19] J. Nidamanuri, C. Nibhanupudi, R. Assfalg, and H. Venkataraman, "A progressive review: Emerging technologies for ADAS driven solutions," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 2, pp. 326–341, Jun. 2022, Conference Name: IEEE Transactions on Intelligent Vehicles, ISSN: 2379-8904. DOI: 10.1109/TIV.2021.3122898.

[20] A. Singh, *Vision-RADAR fusion for robotics BEV detections: A survey*, 2023. DOI: 10.48550/ARXIV.2302.06643. Accessed: Feb. 21, 2025. [Online]. Available: https://arxiv.org/abs/2302.06643.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers*, May 28, 2020. DOI: 10.48550/arXiv.2005.12872. Accessed: Feb. 21, 2025. [Online]. Available: http://arxiv.org/abs/2005.12872.

[22] Y. Wang, V. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, *DETR3d: 3d object detection from multi-view images via 3d-to-2d queries*, Oct. 13, 2021. DOI: 10.48550/arXiv.2110.06922. Accessed: Jan. 24, 2025. [Online]. Available: http://arxiv.org/abs/2110.06922.

[23] W. Ljungbergh et al., *GASP: Unifying geometric and semantic self-supervised pre-training for autonomous driving*, Mar. 19, 2025. DOI: 10.48550/arXiv.2503.15672. Accessed: Aug. 13, 2025. [Online]. Available: http://arxiv.org/abs/2503.15672.

[24] H. Caesar et al., *nuScenes: A multimodal dataset for autonomous driving*, 2019. DOI: 10.48550/ARXIV.1903.11027. Accessed: Sep. 9, 2025. [Online]. Available: https://arxiv.org/abs/1903.11027.

A

Appendix

Here, the following results from Section 4.1.4 are presented. The graphs showcase the evolution of each individual fine-tuning metric dependent on the data transformation method used. For the condensed result of all these graphs please consult tab. 4.1.

A.1 Pre-training Loss Scaling - Expanded Result

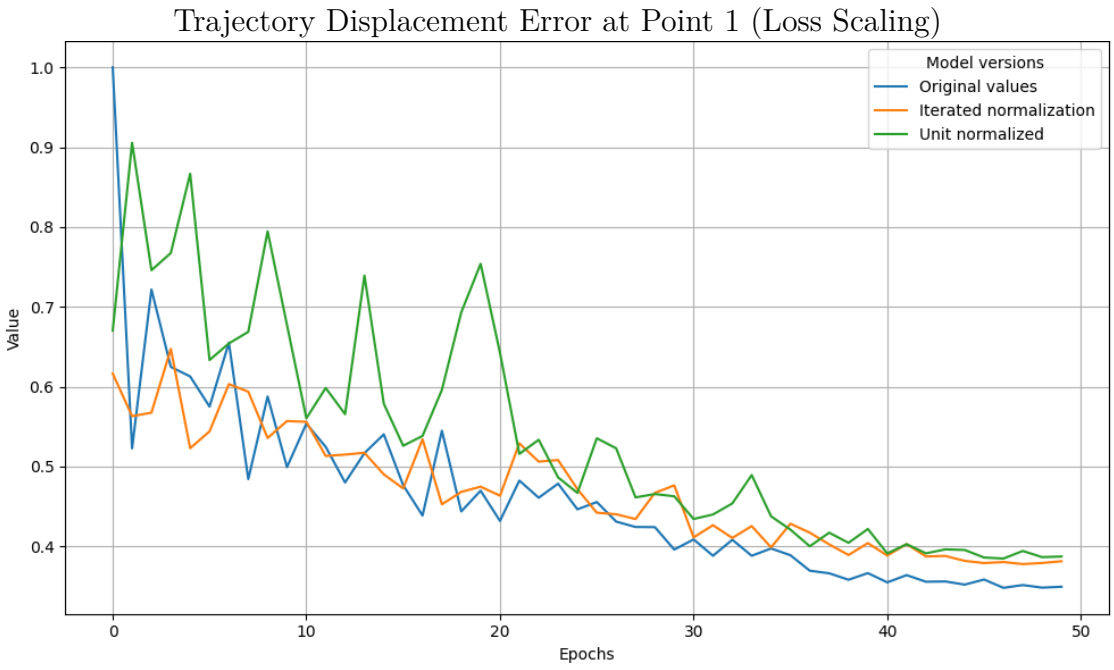


Figure A.1: *Normalization methods evaluated on the metric TDE at point 1.*

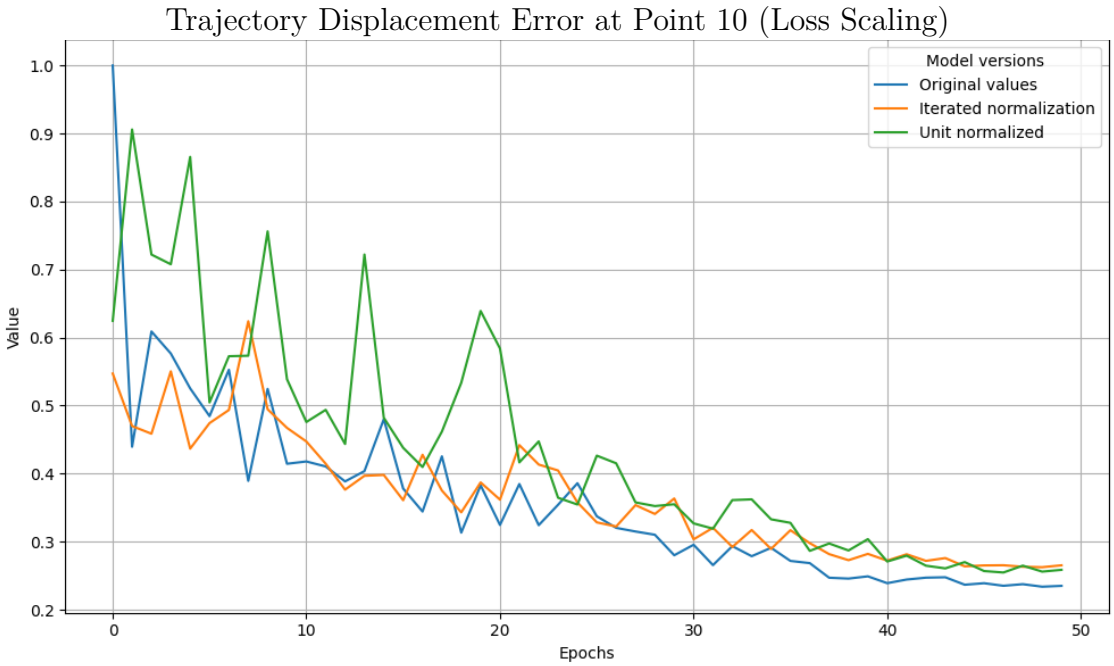


Figure A.2: *Normalization methods evaluated on the metric TDE at point 10.*

Figure A.3: *Normalization methods evaluated on the metric TDE at point 50.*

Figure A.4: *Normalization methods evaluated on the metric AP.*

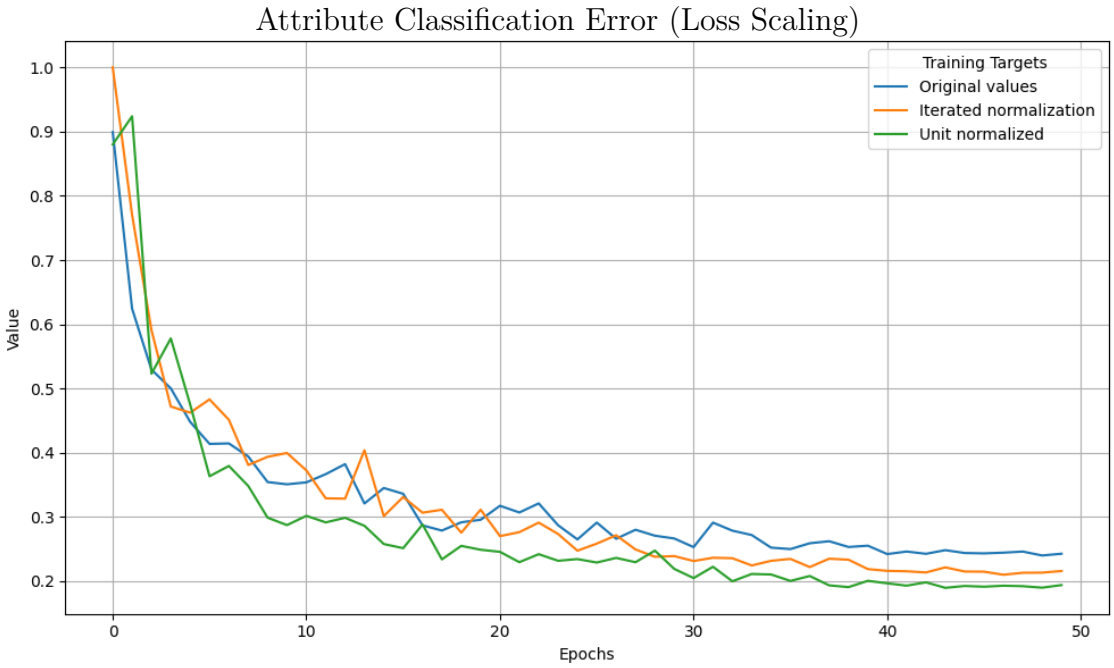


Figure A.5: *Normalization methods evaluated on the metric ACE.*

Figure A.6: *Normalization methods evaluated on the metric AOE.*

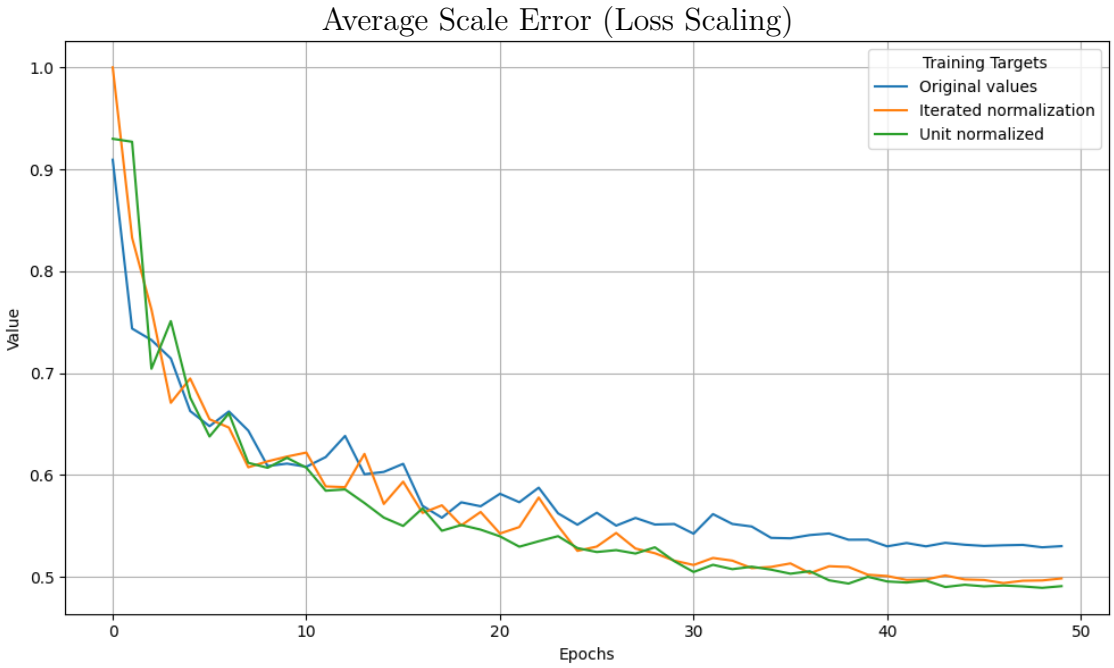


Figure A.7: *Normalization methods evaluated on the metric ASE.*

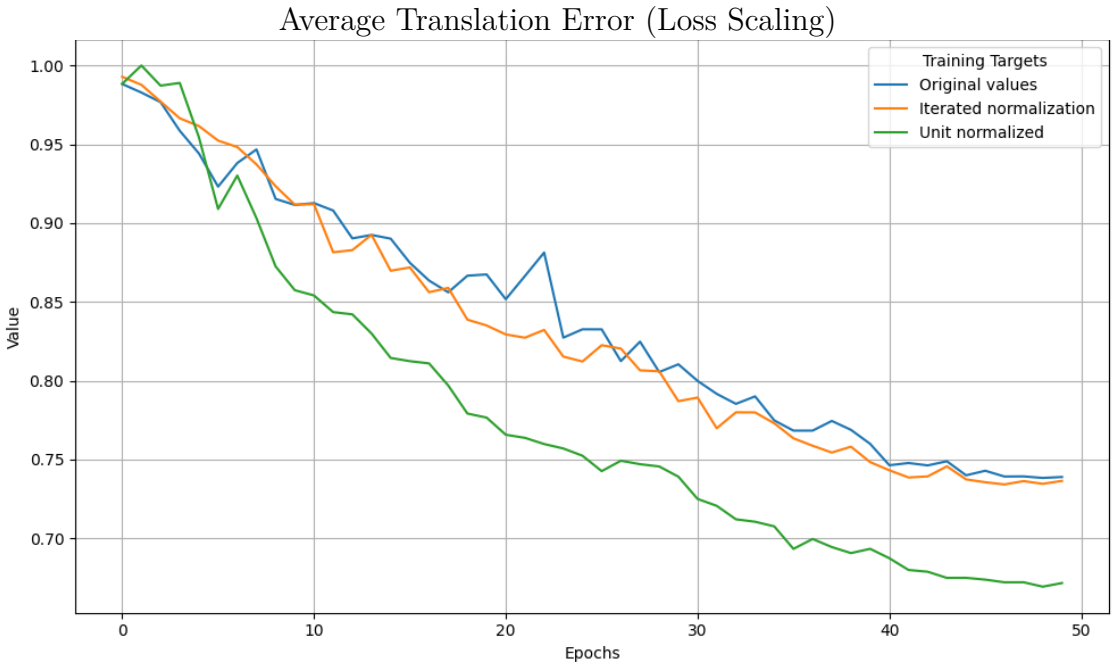


Figure A.8: *Normalization methods evaluated on the metric ATE.*

Figure A.9: *Normalization methods evaluated on the metric IDS.*

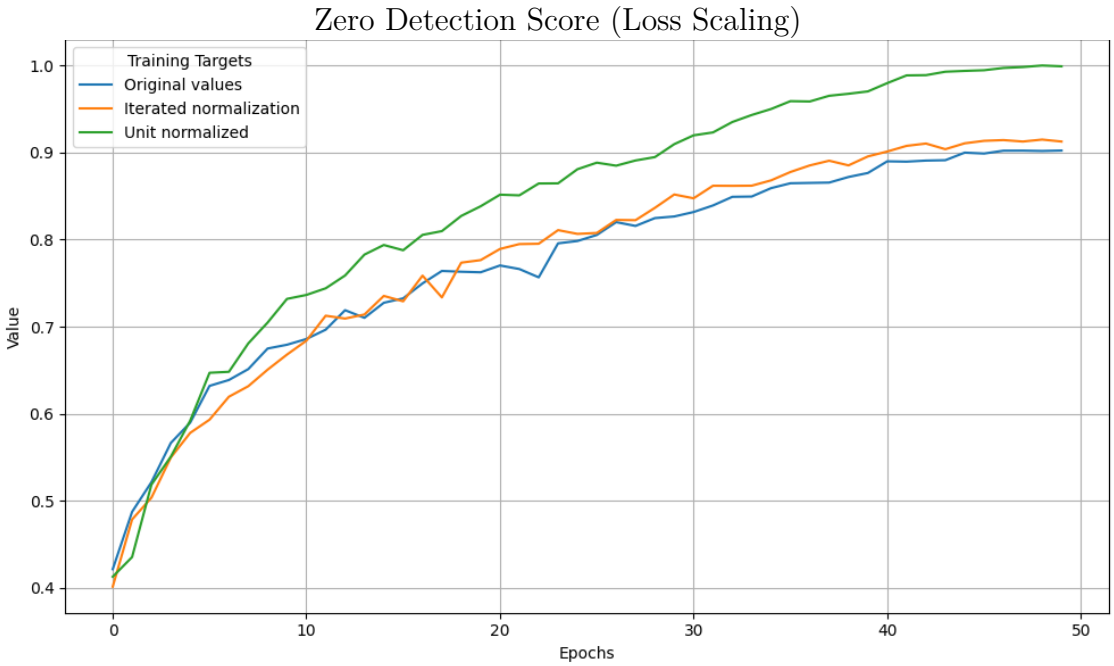


Figure A.10: *Normalization methods evaluated on the metric ZDS.*

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY