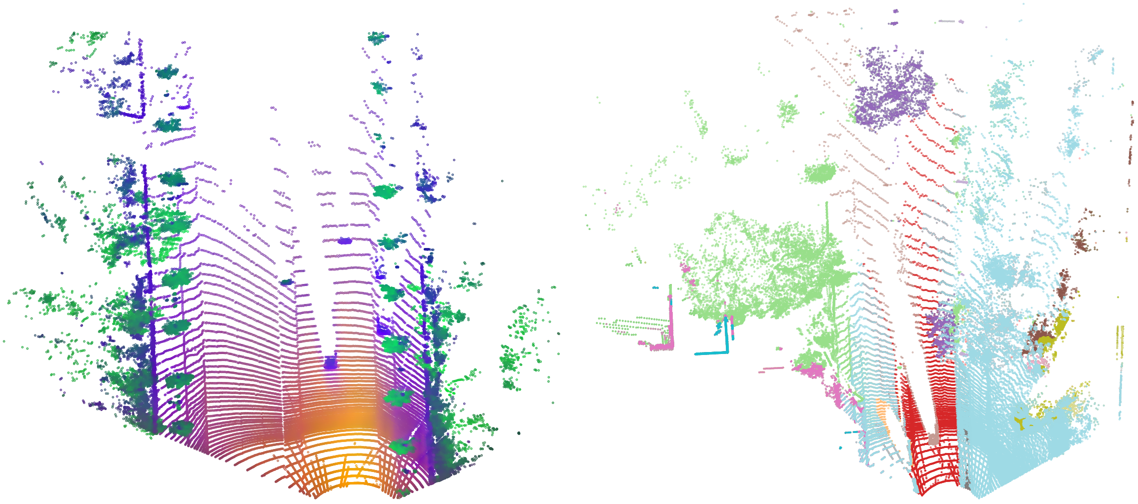
CHALMERS
UNIVERSITY OF TECHNOLOGY



Self-Supervised Pre-Training with Vision Foundation Models on 3D Point Clouds

Comparing Downstream 3D Object Detection Performance Across Varying Label Quantities for Two Pre-training Strategies

Master's thesis in Electrical Engineering

Anni Carlén and Pauline Nässlander

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's thesis 2025

Self-Supervised Pre-Training with Vision Foundation Models on 3D Point Clouds

Comparing Downstream 3D Object Detection Performance Across Varying Label Quantities for Two Pre-training Strategies

Anni Carlén and Pauline Nässlander

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

Self-Supervised Pre-Training with Vision Foundation Models on 3D Point Clouds
Comparing Downstream 3D Object Detection Performance Across Varying Label Quantities for Two Pre-training Strategies
Anni Carlén and Pauline Nässlander

Supervisors: Maryam Fatemi, Zenseact AB
Willem Verbeke, Zenseact AB
Mahan Rafidashti, Zenseact AB

Examiner: Lars Hammarstrand, Department of Electrical Engineering,
Chalmers University of Technology

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

Cover: Predicted pseudo-labels from lifted DINOv2 features (left) and SAM 2 segmentation masks (right).

Typeset in LaTeX
Gothenburg, Sweden 2025

Self-Supervised Pre-Training with Vision Foundation Models on 3D Point Clouds
Comparing Downstream 3D Object Detection Performance Across Varying Label Quantities for Two Pre-training Strategies
Anni Carlén and Pauline Nässlander
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The ability to accurately identify objects and their spatial properties in 3D is a critical task within the field of autonomous driving. To achieve this goal, deep neural networks identify objects and determine their corresponding 3D position, extent and orientation. For deep models to learn general properties of the environment and be able to identify objects, large amounts of labelled data are required. Since labelled data is expensive and time-consuming to obtain, it is crucial to reduce reliance on it without compromising performance. Therefore, this thesis investigates two pre-training tasks for 3D object detection in LiDAR point clouds and highlights their effectiveness in one-shot and few-shot detection. The pre-training methods in this project utilise two different web-scaled foundation models, namely, SAM 2 and DINOv2. As both models are trained on 2D images, we lift their outputs to 3D to generate pseudo-labels for training a 3D object detection model in a self-supervised fashion. Predicted features from DINOv2 are used in the first pre-training task, while segmentation masks from SAM 2 are used in the second task. As a result, the performance of each method reflects both the choice of pseudo-labelling method and the quality of the underlying foundation models.

The results show that the model effectively learns to identify objects from a single annotated sample after pre-training on either SAM 2 or DINOv2. When only a few samples are available, the pre-trained and fine-tuned models significantly outperform the baseline which is trained fully-supervised from scratch. This emphasises how these pre-training methods enabled the model to learn and recognize less common object categories even when they are represented by only a few instances in the dataset.

Keywords: object detection, LiDAR, foundation models, self-supervised, pre-training, point cloud, thesis.

Acknowledgements

We would like to express our sincere thanks to our supervisors Maryam Fatemi, Willem Verbeke, and Mahan Rafidashti at Zenseact, for their invaluable guidance, support, and insightful feedback throughout the thesis and the development of the project. We are truly grateful for your involvement.

We are also deeply grateful to our academic supervisor at Chalmers, Lars Hammarstrand, for his valuable perspectives and especially for his insights when defining the scope and structure of the project.

A special thanks to Willem Verbeke for his consistent support and valuable discussions without which this project would not have reached its full potential. We recognise the time and effort you have invested in helping us carry out the project.

Anni Carlén & Pauline Nässlander, Gothenburg, 2025-06-17

Contents

Acronyms

AABB	Axis-Aligned Bounding Boxes. 38, 54
ADAS	Advanced Driver Assistance Systems. 1
AI	Artificial Intelligence. 8
BCE	Binary Cross Entropy. 14, 36, 37
BEV	Bird Eye's View. 5, 9, 10, 13, 19–21, 24, 26, 27, 38, 54, I
CNN	Convolutional Neural Network. ix, 4, 5, 10, 11, 17, 19–21
DBSCAN	Density-Based Spatial Clustering of Applications with Noise. 55, 56
DETR	Detection Transformer. 4
DLA	Deep Layer Aggregation. 4, 11, 20
DVT	Denoising Vision Transformers. 26, 27
EMA	Exponential Moving Average. 2
FFN	Feed Forward Network. 9, 27, 28, 31
FN	False Negatives. 38
FP	False Positives. 38, 54
GPU	Graphical Processing Unit. 8, 41, 42, 46
GT	Ground Truth. 13, 14, 17, 28, 36, 38, 42, 43, 45, I–IV
IDA	Iterative Deep Aggregation. 11
IoU	Intersection over Union. 38, 39, 54, I
IQR	Inter Quartile Range. 46
LiDAR	Light Detection and Ranging. 1–3, 6, 7, 13, 17–20, 26, 28, 31, 34, 42, 53, 57

MAE Mean Absolute Error. 13, 14
mAP Mean Average Precision. 38, 39, 41, 44, 46, 49, 54–56
MPM Masked Point Modelling. 6
MSE Mean Squared Error. 13, 14, 31

NLP Natural Language Processing. 2, 6, 8
NMS Non-Maximum Suppression. 24, 32, 33, 54, I

PCA Principal Component Analysis. x, xv, 14, 27, 41, 42

R-CNN Region-based Convolutional Neural Network. 3
ResNet Residual Network. ix, 4, 10, 20
RPN Region Proposal Network. 5

SAL Segment Anyting in LiDAR. 7
SAM Segment Anything Model. 2, 3, 7, 32–34, 54–56

TP True Positives. 38, 54

ViT Vision Transformers. 2

YOLO You Only Look Once. 3

ZOD Zenseact's Open Dataset. 7, 17, 18, 26, 27, 33, 39, 41, 42, 46, 49, 54

List of Figures

List of Tables

1

Introduction

Over the past year (2024), 210 people have lost their lives due to traffic accidents involving one or more vehicles in Sweden [1]. This toll persists despite road safety being highly prioritised by Swedish authorities, which has made Sweden one of the safest countries in the world for traffic [2]. Furthermore, human error has been identified to be a frequent cause of traffic accidents. In the USA, it is estimated to be the critical reason (i.e. the immediate cause or last failure in the chain of events) in 94% of total car crashes [3]. Of these 94%, approximately 41% of crashes are broadly attributed to perception errors made by the driver. Self-driving cars and Advanced Driver Assistance Systems (ADAS) have emerged as ways to decrease human error in traffic accidents and have the potential to substantially lower traffic accident rates and save lives.

The safety of self-driving systems can only be guaranteed if the car's ability to understand the surroundings is robust, as everything from trajectory-planning to emergency braking systems relies on the output from these perception systems. One way to provide these systems with sensible data about their surroundings is to use object detection. This provides class probabilities, position, size and orientation of all identified objects in the perceptive field. Moreover, the perception of the car's surroundings needs to be resilient towards different weather conditions affecting visibility and/or to sensor failures. This may be achieved by employing computer vision systems relying on different sensor modalities for predictions. For instance, while the Light Detection and Ranging (LiDAR) sensor is robust to challenging lighting conditions, cameras often suffer in those settings [4]. On the other hand, LiDAR is prone to inaccuracies in snow or rain due to changes in the reflection of laser signals, an area where cameras often can provide more reliable information. Therefore, object detection from LiDAR data remains a valuable tool for ensuring road safety, particularly when combined with other sensor modalities.

LiDAR based object detection in 3D normally requires large amounts of annotated data to enable the model to learn all the important features for identifying objects in a point cloud [5]–[8]. These annotations are very expensive to obtain as they require humans to manually perform the detection task when annotating the point clouds. This is more complex than the annotation of images since humans are less used to identifying objects in a point cloud setting. As a result, the large amounts of 3D point cloud annotations needed for training a robust and detailed model are hard to come by. Fully supervised approaches rely heavily on these labour-intensive annotations, which make scaling difficult and costly. However, the identification of objects in images is a well-researched field which has produced several large and accurate foundation models as powerful tools for both object detection and semantic segmentation [9]–[11]. This project will therefore develop pre-training strategies utilizing Vision-based foundation models in the 2D image domain and lift their output to a LiDAR point cloud for self-supervised training of a 3D object detection model on LiDAR data with the goal of reducing reliance on expensive annotations.

1.1 Background

This thesis aims to develop pre-training methods for an object detector in LiDAR point clouds. To provide a fundamental understanding of relevant topics, this chapter will present several methods that effectively perform 3D object detection and techniques which have been developed to reduce annotated data requirements while still maintaining strong performance.

1.1.1 Vision-based foundation models

Foundation models within the field of computer vision enable broad generalization across diverse tasks. These models can be incorporated in pre-training architectures that can be fine-tuned with minimal data for a specific task [12]. However, since these models often achieve high accuracy on general datasets they can also be used directly, without fine-tuning, to facilitate downstream tasks, as demonstrated in [13], [14]. Two such vision based foundation models are DINOv2 [11] and Segment Anything Model (SAM) [10].

DINOv2
DINOv2 is a self-supervised vision transformer model from Meta AI that generates high-quality, general-purpose visual features, enabling effective performance across diverse image and pixel-level tasks without requiring fine-tuning [11]. DINO is a self-**di**stillation strategy with **no** labels and was designed to adress some of the limits to Vision Transformers (ViT) such as their large data requirement and expensive computations by performing extensive self-supervised pre-training as successfully used in Natural Language Processing (NLP) [15]. This makes the DINOv2 model highly scalable as it does not require manual labels. They use a teacher-student framework where the teacher is built on an Exponential Moving Average (EMA) of past iterations of the student to enable the self-supervision.

DINOv2 is shown to perform well on a wide range of benchmarks, closing the performance gap with weakly supervised alternatives. The model demonstrates strong capabilities in understanding object parts and scene geometry [11].

SAM
SAM is another foundation model by Meta AI which, similarly to DINOv2, shows impressive generalization across diverse domains [10]. SAM is a promptable segmentation model specifically developed to address the challenge of generalization across

diverse segmentation tasks in computer vision, demonstrating zero-shot performance that is often competitive with, or even superior to, fully supervised models. SAM consists of a heavy image encoder, a prompt encoder and a more lightweight mask decoder and was trained on a labelled segmentation dataset of over 1 Billion masks on 11 Million images to obtain its performance. Improving on this work, Meta AI releases SAM 2 [16] which is more accurate and 6 times faster than SAM on image segmentation.

The above-mentioned models are both on the cutting edge of vision-based foundation models and are highly suitable for the application area of this project. Therefore, it is of great interest to compare the two in this context to determine which one provides a better pre-training signal for optimizing the performance of a LiDAR-based object detector.

1.1.2 Object detection in 2D

To grasp the fundamental requirements, capabilities, and limitations of object detection in 3D, it is useful to first examine the advancements and methodologies in 2D object detection.

Region Proposal

Region proposal is a method which reduces the object detection task to a classification problem by generating a limited set of candidate regions that likely obtain an object [17]. Among these proposals, many will be noisy, but some will accurately bound the objects of the original image. Methods like selective search [18] group similar pixels based on similarities, producing around 2,000 proposals per image. Having a limited set of proposed regions make the detection process more efficient compared to sliding window approaches where the entire image is treated as possible regions and need evaluation. This approach balances high recall with computational efficiency and has been foundational in the development of region-based object detectors such as Region-based Convolutional Neural Network (R-CNN) proposed in [19].

Single-Stage with Post-Processing

In contrast to the region proposal approach to object detection, one-stage networks with post processing gained attention due to their efficiency and simplicity relying solely on convolutional operations. You Only Look Once (YOLO) [20], is perhaps one of the most influential single-stage detection models achieving real-time speed and competitive accuracy. Several advancements have been made to the detector framework showing its potential, including [21], [22]. In the early versions, the YOLO method relies on anchor box generation to provide a starting point for bounding box generation by regressing offsets from these anchors instead of predicting boxes from scratch. The anchor proposals are applied on feature maps and properties such as the anchor box sizes and aspect ratios are manually selected based on heuristics or knowledge about the dataset. The FCOS model (Fully Convolutional One-Stage) presented in [23] further advanced the single-stage approach by introducing an anchor-free framework that directly predicts object locations and sizes from obtained feature maps.

Eliminating the reliance on prior predictions or anchor proposals in the object detection pipeline offers several advantages [22]–[24]. Firstly, it simplifies the overall detection process by removing the need to carefully tune hyperparameters associated with anchor box generation, which is crucial in order to obtain competitive performance. Moreover, the anchor generation procedure is both computationally demanding and time consuming and the training procedure of the subsequent network typically demands a larger training memory footprint compared to anchor-free versions. Additionally, anchor-based detectors often suffer difficulties handling object candidates with significant shape and size variations [21], [23].

Another anchor-free approach is CenterNet [25] which views the object detection problem as a keypoint estimation problem. Each ground truth object is assigned a centre point and then they use keypoint estimation to find and classify the centre points while the other object properties such as size and orientation are regressed. They use several different fully-convolutional encoder-decoder networks built on previous successful keypoints estimation networks such as Hourglass [26], ResNet [27] and Deep Layer Aggregation (DLA) [28]. The regression could also be used to predict 3D properties from the 2D images further showing the strength in representing objects as points.

End-to-End
In contrast to the single-stage methods an end-to-end detector requires no post-processing to obtain the final bounding box predictions. A transformer-based model for object detection was first introduced in [24] which suggested an end-to-end method for object detection that improved the efficiency of the detection pipeline. The framework called Detection Transformer (DETR) uses an encoder-decoder architecture to output several parallel predictions based on the relationship between objects and global image context. DETR demonstrates comparable accuracy and run-time performance to the well-established and highly optimized Faster R-CNN baseline on the COCO object detection dataset. DETR performs especially well on large objects in comparison to the faster R-CNN, which is possibly due to the self-attention in the transformer structure which could provide the model with the ability to process global information.

After evaluating various combinations of different foundation models and detection architectures, [29] shows that using the pre-trained DINOv2 model for the vision backbone with DETR detection architecture achieves state-of-the-art performance in few-shot object detection. Similarly to [24], they present issues with detecting small objects as well as detecting objects in dark environments.

1.1.3 3D Object Detection

In order to effectively evaluate methods for implementing a 3D object detector it is important to first gain a deeper understanding of existing methods in the field. By analysing this we can better identify the most effective solutions for our application.

Backbone
Several 3D object detection models for point clouds utilize a 3D backbone, typically

VoxelNet [30], [31] or PointPillars [32], to encode point cloud data into feature vectors. These encoded features are then processed using either CNNs or a transformer-based architecture [33]–[35]. VoxelNet divides the point cloud into small 3D voxels, they then generate per-voxel-features followed by 3D convolutions [30]. Improving upon VoxelNet, the authors of [31] proposed SECOND which incorporates sparse convolutions that operate only on occupied voxels. This addition significantly reduces the computational complexity of VoxelNet. In contrast, PointPillars, while inspired by VoxelNet, improves efficiency by replacing 3D voxels with vertical pillars. It applies 2D convolutions in Bird Eye's View (BEV) space instead of 3D convolutions, further reducing computational complexity. Both approaches enhances model efficiency by effectively compressing the point cloud representation while minimizing information loss, which could also be beneficial for this project.

CNN based

Building upon the introduction of VoxelNet and PointPillars as 3D backbones, these architectures are extended in their respective papers [30], [32] and serve as complete CNN-based 3D object detectors on point clouds. Both VoxelNet and PointPillars integrate their respective encoding methods with Region Proposal Network (RPN)s to perform end-to-end 3D object detection. VoxelNet uses multi dimensional convolution layers followed by 2D convolutions on the encoded voxel-wise features to generate a dense feature map as input to the RPN [30]. PointPillars, on the other hand, scatters its pillar-based encodings into a pseudo-image which is processed by a regular 2D CNN and finally sent through an RPN [32].

In contrast to the above networks, the two-staged 3D detector PointRCNN operates directly on raw points [36]. PointRCNN consists of a bottom-up stage for 3D proposal generation by segmenting the foreground points and generating proposals from these. The second stage is a refinement network which transforms the proposals to canonical coordinates for accurate box prediction. Published in 2019, PointRCNN outperformed VoxelNet[30] and SECOND[31] on several classes on the KITTI dataset [36] at the cost of efficiency due to dealing with the large and sparse data in the raw point cloud [37].

Another CNN based 3D object detector called CenterPoint was proposed in [33]. CenterPoint utilizes a CNN-based architecture to detect centres of objects and regresses 3D bounding box properties such as size and orientation. CenterPoint, puplished in 2021, outperforms earlier results on several benchmarks and demonstrates the effectiveness of representing 3D objects as points rather than boxes. This project draws inspiration from CenterPoint by adopting a similar one-stage detection approach for 3D object detection. This single-stage method is well-suited for use in embedded systems due to it's simple design and efficiency. Therefore, this thesis will take inspiration from this approach to create an object detection base model which is applicable in autonomous driving.

Transformer based

Several 3D object detectors have been implemented using transformer-based backbones [6], [8], [38].

One approach is to use end-to-end transformer models for 3D object detection, such as 3DETR [38] (based on the model presented in [24] for 2D object detection). The model processes raw 3D point clouds directly. Since the model does not divide the point cloud into grids, it preserves details but is also more computationally heavy compared to grid-based approaches such as VoxelNet [30] and PointPillars [32]. This is due to the quadratic complexity of the self-attention mechanism.

Point-BERT [6], based on BERT [39], is a transformer-based model designed for 3D point clouds that utilize Masked Point Modelling (MPM) as a pre-training strategy. Inspired by masked token prediction in NLP, point-BERT randomly masks patches of a point cloud and learns to reconstruct them, forcing the model to capture meaningful geometric structures and spatial relationships. The authors emphasizes the effectiveness of self-supervised learning for point cloud understanding and the ability to generalize across different 3D tasks.

1.1.4 Self-supervision in 3D

To reduce the reliance on large-scale annotated datasets, self-supervision has emerged as a powerful approach. Understanding its role in 3D object detection and segmentation will highlight its potential in learning meaningful representations.

Self-supervised object detection in 3D
Several previous works on self-supervised training for object detection rely on motion to identify unlabelled objects in 3D (RGB-D) sequences [40]–[42]. These motion-based approaches have proven efficient in locating moving objects with little to no labels. However, they fail to provide predicted semantic labels such as "car", "pedestrian" etcetera.

Another approach is presented in [43], which during training utilizes a calibrated camera-LiDAR setup and per-point compressed CLIP [9] features to provide semantic labels. It then leverages motion data from point clouds, matching it with 2D image-text pairs to classify objects. While this effectively assigns labels to detected objects, the training requires motion data sequences to identify them. In contrast, our approach will take the distillation of web-based foundation model outputs one step further by creating pseudo-labels that incorporate objectness and positional context as a pre-training signal for an object detector.

Furthermore, [8] uses an encoder-decoder architecture and self-supervised pre-training on 2D images to learn 3D representations in point clouds. An encoder is fed with a masked point cloud. Using multiple blocks with self-attention layers, the model learns global 3D structures. After encoding, the visible tokens are combined with a set of learnable masked tokens, which are input into the decoder. The decoder learns to reconstruct the missing 3D coordinates.

To tackle the challenge of the limited size of 3D object detection datasets compared to many 2D datasets, [5] suggested a mining procedure of 3D box data without semantic labels. They generated two dimensional instance segmentations using an open source panoptic segmentation model and backprojected these instance proposals to a 3D bounding box using certain assumptions about objects shapes and orientations. A detector was then trained to produce these mined annotations from LiDAR point clouds. The results showed that with enough mined data the model outperformed a model trained on a human-annotated training set despite the noisy nature of the mined bounding boxes. This thesis will aim to leverage the shown potential of derived labels from 2D images shown in [5] while improving the quality of the pseudo-annotations by directly unprojecting SAM segmentation masks or DINOv2 features onto the LiDAR point cloud.

Self-supervised 3D point cloud segmentation
Recent research within self-supervised LiDAR point cloud segmentation utilizes the distillation of foundation models to generate pseudo-labels lifted to the 3D domain to train a zero-shot 3D segmentation model called Segment Anyting in LiDAR (SAL) [13]. This novel approach tackles the task of panoptic segmentation with a zero-shot model trained on pseudo-label-masks generated by utilizing a calibrated camera-LiDAR setup to lift segmentation masks from SAM to 3D. Similar to [43], this paper utilizes CLIP [9] for semantic distillation. However, in contrast to the previous approach, [13] extracts the CLIP features per segmented object instance to gain a more object-centric semantic understanding. The proposed pseudo-label-engine and trained model achieves promising results on several benchmarks, indicating that the model has obtained a sense of objectness in 3D LiDAR point clouds from the lifted 2D segmentations. This thesis will take inspiration from this work and use outputs from vision-based foundation models lifted from a camera image to an associated LiDAR point cloud as a pre-training signal for a LiDAR-based object detector.

1.2 Societal, Ethical and Ecological Aspects

One of the primary ethical aspects of this project is ensuring road safety. To ensure that self-driving vehicles safely navigate an unpredictable world, it is essential to have a robust computer vision system. Specifically, reliable object detection is an important tool as it is a computationally effective way to describe important visual features for a computer. High-performing object detection is important in many self-driving applications, such as emergency braking, where reliable and fast vision systems are vital for quick decision-making.

When training such networks, it is of great importance to ensure inclusiveness and diversity in the labelled data sets, this is to ensure robust models and reliable and fair decision-making [44]. Another aspect related to data is data retrieval, privacy and annotation methods. When it comes to privacy, Zenseact's Open Dataset (ZOD) uses blurring to protect the identities of individuals in photos, addressing privacy concerns.

The goal of this project is to reduce the reliance on manually labelled data. Achieving this would not only accelerate the development of autonomous driving systems but also enhance their ability to detect objects accurately with smaller, more efficient datasets. Ultimately, efforts within this field contribute to safer and more reliable self-driving cars while reducing the manual labour imposed by ground truth annotation.

Another ethical aspect to consider when developing Artificial Intelligence (AI) models is the emissions of carbon dioxide from large Graphical Processing Unit (GPU) clusters. While this concern is especially evident when training large NLP models, which have grown exponentially in size, similar principles apply to foundation models across other domains. According to [45] the training of GPT-3 consumed approximately 1,287 MWh of energy and the estimated emissions are 552 tons of carbon dioxide. In addition, the carbon footprint of training a single large language model can be comparable to lifetime emissions of several cars [46]. However, [45] also states that one environmental benefit of large models is the ability of few-shot generalization without retraining, placing their initial carbon cost in comparison to repeatedly training smaller, task-specific models

A different ecological aspect to consider is the significant processing power self-driving cars require, leading to increased carbon emissions [47]. Research suggests that the adoption of autonomous vehicles could accelerate the issue of greenhouse emissions. To handle this problem, it is crucial to develop more efficient hardware as well as neural networks and algorithms. On the other hand, another study suggests that the availability of electric autonomous cars can impact the population's choices of commuting in larger cities, toward more sustainable options [48]. Specifically, with a change in today's transportation industry, it is possible to decrease greenhouse emissions by up to 34% by 2050. Overall, it can be argued that the benefits, such as the potential to reduce traffic accidents and save lives outweigh these downsides. Within the scope of this project, our responsibility is to produce as effective algorithms as possible to ensure more efficient processing and use of resources.

2

Theory

This chapter offers deeper insights into the theoretical background necessary to gain an understanding of the methods presented later. It further delves into the technical details needed to obtain an understanding of this thesis methodology.

2.1 PointPillars Voxalization

PointPillars provide an efficient technique for encoding 3D point clouds into a BEV feature representation or pseudo image [32]. The procedure is illustrated in Figure 2.1.

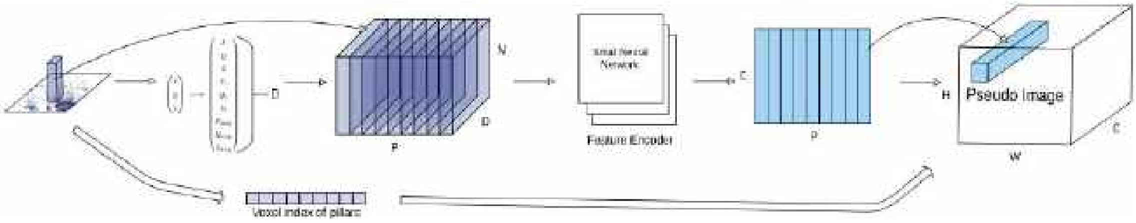


Figure 2.1: Pipeline of voxalization using PointPillars as described in [32].

The point cloud is first processed pointwise, then voxelised into pillars which are stacked to form a pseudo image. The 3D space is divided into a grid in the BEV plane that extends through the entirety of the third (height) dimension dividing it up into pillar-like cells. The points that fall into any of these cells form the voxels (or pillars).

Each point, given by it's x, y, z coordinates and it's reflectance r is augmented into a vector of length $D = 9$ which includes the original point as well as the offset to the centre of the pillar it falls in (x_p, y_p) and the distance to the average of the all points within the same pillar (x_c, y_c, z_c). The point vectors of length D are then stacked and zero-padded into a tensor of shape (D, P, N) where P is the number of populated pillars and N is the maximum number of points per pillar. If a voxel includes more points than N, random sampling is used. The grid location of each voxel is kept for later use. A single layer Feed Forward Network (FFN) is applied pointwise which generates a tensor of shape (C, P, N) followed by a max operation over channels which produces a (C, P) size tensor, incorporating the spatial information of the point distribution within each pillar.

The learned features are subsequentially converted into a 2D pseudo-image of shape (C, H, W), where H,W are the height and width of the grid in the BEV plane and C is the length of the encoded feature vectors. This step is done using the list of saved grid index of each pillar.

2.2 ResNets

ResNet leverages shortcut connections across one or more layers to form residual blocks, which are then stacked sequentially to build deep neural networks [27]. This method effectively addresses the issue of vanishing gradients which is a common concern when training deep networks. By adding a shortcut connection over layers, ResNets allow the network to learn residual mapping, which essentially is the difference between input and the desired output, instead of learning the full transformation between input and output. Implementing residual blocks allow for deeper networks without reduced performance and the network can still be trained end-to-end during back-propagation.

ResNet demonstrates impressive results in image recognition tasks, particularly in scenarios where networks would otherwise degrade in performance due to over-fitting or optimization difficulties. It remains an effective method for training very deep convolutional networks, making it highly relevant for applications in computer vision, where large-scale deep learning models are essential for tasks such as object detection, segmentation, and classification. Today, ResNet is the standard architecture for any CNN-based computer vision model.

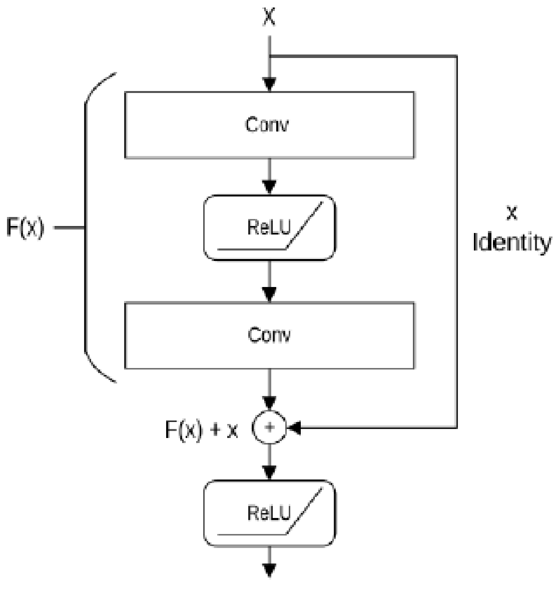


Figure 2.2: Architecture of a Residual Block adapted from [27].

2.3 Deep Layer Aggregation

DLA is a method designed to improve CNNs by aggregating information from multiple layers throughout the network [28]. This technique enables models to achieve better accuracy while also reducing the number of parameters. With DLA, it is possible to maintain resolution while simultaneously gaining a richer understanding of various features and improve efficiency. In other words, DLA allows for the advantages of both deep and shallow convolutions to be leveraged, enabling the model to both pinpoint the object positions using fine details from shallower layers, while using rich features from deeper layers to confirm the object class. This ability is especially important for the object detection task.

Iterative Deep Aggregation (IDA), shown in Figure 2.3, is a type of DLA designed to fuse resolution and information density [28]. IDA begins aggregation at the shallowest, largest resolutions, and iteratively merges deeper layers with lower resolutions. This progressive aggregation allows shallow features to be refined as they move through the network and allows for information from shallower layers to be merged with higher level features.

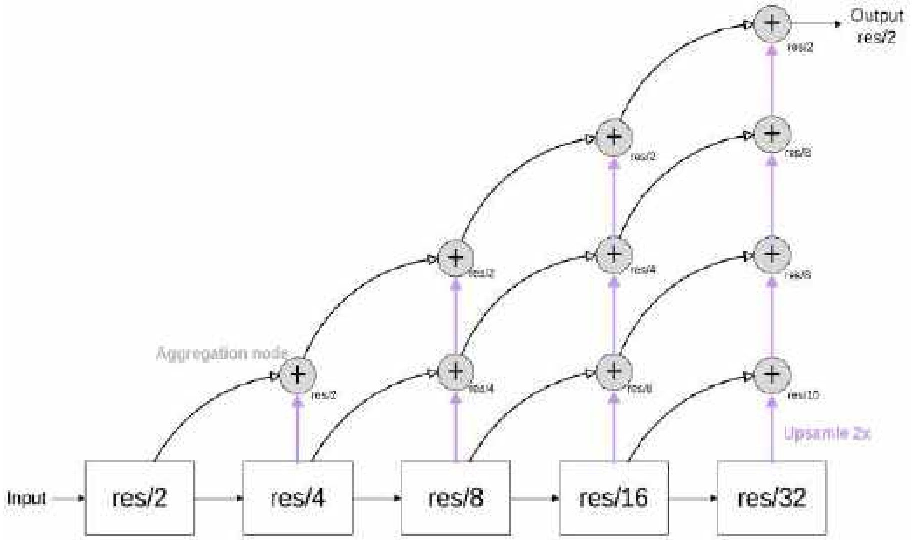


Figure 2.3: Structure of Iterative Deep Layer Aggregation adapted from [28]. The deeper layers are upsampled to 2 times its resolution to match the the shallower layer resolution using interpolation. The shallower layers and upsampled deeper layer are combined in an aggregation node consisting of a convolution, batch norm and a non-linear activation function.

2.4 CenterNet

The CenterNet paper [25] presents an object detection method for camera images which leverages keypoint estimation to identify object centre points while simultaneously regressing object properties such as size. CenterNet can be extended to

3D object detection in 2D camera images by also incorporating regression of 3D attributes like depth and orientation.

For 2D object detection on an image with Height H and width W, the CenterNet produces 3 output predictions. The first one is a keypoint heatmap $\hat{Y}$ of size (H/R, W/R, C) where C are the number of classes, and R is the output stride. The heatmap contains numbers between 0 and 1 where large values at position (x, y, c) represents the model being more confident that there is an object of class c with its centre point in the receptive field of the (x, y) cell. Additionally, CenterNet regresses a local offset $\hat{O}$ and object size $\hat{S}$ both of size (H/R,W/R,2). The offset contains the $(\delta x, \delta y)$ offsets of the object centerpoint from the middle of the cell and the size (h, w) is the height and width of a potential object with its centerpoint in the given cell.

CenterNet uses a single network to predict a total of $C + 4$ outputs at each location. The outputs $\hat{Y}, \hat{O}, \hat{S}$ share a fully convolutional backbone and then for each property is passed through a convolutional chain consisting of a 3×3 convolution followed by a ReLU layer and another 1×1 convolution see Figure 2.4.

The paper uses a Gaussian kernel in the $x - y$ plane on the target heatmap to penalize cells nearby true keypoints less than those far away. The paper then utilizes "penalty-reduced pixelwise logistic regression with focal loss"[25] which is given as:

$$L = \frac{-1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc} = 1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases} \tag{2.1}$$

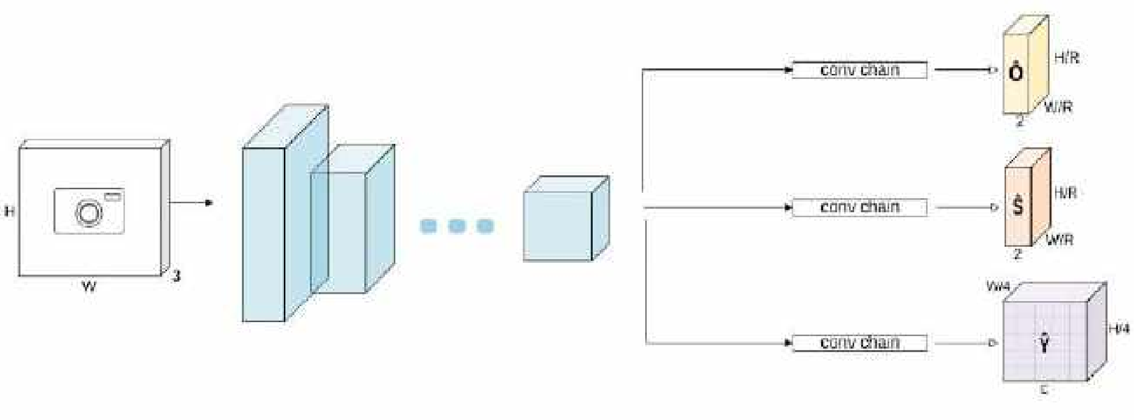


Figure 2.4: CenterNet [25] architecture for 2D object detection on images.

where α and β are tunable hyperparameters set to 2 and 4 respectively in [25] and N is the number of keypoints in the image. They further train the offset and size with an L1-loss (described in Equation 2.3) only applied at locations where the heatmap is non-zero, and ignoring all other locations.

2.5 CenterPoint

Tackling the task of LiDAR 3D object detection, [33] presents a similar approach as the CenterNet paper ([25]). They first detect objects as keypoints in the BEV plane in a heatmap trained similarly to CenterNet, then they regress object properties such as offset refinement, height above ground, 3D size and orientation. The orientation is described by two parameters ($\sin(\varphi)$, $\cos(\varphi)$), where φ is the yaw angle.

2.6 Loss functions

Below is a description of the different loss functions and algorithms used to calculate the difference between ground truths and predictions in this work.

2.6.1 Mean Squared Error

Mean Squared Error (MSE), also known as L2loss, is a widely used loss function for regression problems [49]. The error between the actual value (y), called Ground Truth (GT), and the prediction (x) is calculated as the mean squared error:

$$f_{mse}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{2.2}$$

MSE is sensitive to outliers, as the loss function heavily penalizes larger errors [49].

2.6.2 SmoothL1

SmoothL1loss is based on L1loss, also called Mean Absolute Error (MAE), and is defined as:

$$f_{mae}(x, y) = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i| \tag{2.3}$$

where x is the prediction and y is the GT. The MAE indicates how much model predictions differentiates from the real values [49]. MAE-loss is, similarly to MSE-loss, commonly used in regression tasks, but unlike MSE, MAE is more robust to outliers as it penalizes all errors linearly to their size.

SmoothL1loss is defined as:

$$f_{smoothL1}(x) = \sum_{i=1}^{N}\begin{cases} 0.5 \cdot {x_i}^2 & \text{if } |x_i| < \beta \\ \beta \cdot (|x_i| - 0.5\delta) & \text{if } |x_i| \geq \beta \end{cases} \tag{2.4}$$

where x_i is the difference between the ith predicted and GT point and $\beta = 1.0$. This can be seen as a compromise between MAE-loss and MSE-loss [50]. Depending on whether a value is under or over a certain threshold β, it will be treated differently.

Values under β will be treated as quadratic (like MSE), and values over will be treated as linear (like MAE). This means that large errors are penalized less harshly than with MSE, making the loss function more robust to outliers.

2.6.3 Binary Cross Entropy Loss

Binary Cross Entropy (BCE) loss is used to measure the difference between predicted probabilities ($\hat{y}$) and binary target labels (y). The loss function is defined as $f_{bce}(y) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})\right]$. It is commonly applied in binary classification tasks [49]. The output of BCE loss represents the likelihood of a correct classification. Because it uses a logarithmic formulation, BCE loss heavily penalizes predictions that are confidently incorrect. With minimized BCE loss, the likelihood of true classes is maximized.

2.6.4 Dice Loss

Dice loss is designed to measure the overlap between predicted and GT segmentations, making it particularly well-suited for image segmentation tasks [51]. One of the key advantages of dice loss is that it penalizes both false positives and false negatives, which makes it especially useful when working with imbalanced datasets. Dice loss is derived from the dice coefficient, which measures the similarity between two sets of data $\text{Dice Coefficient} = \frac{2\times|\text{Prediction}|}{|\text{Prediction}|+|\text{Target}|}$. The dice loss is computed accordingly $\text{Dice Loss} = 1 - \text{Dice Coefficient} = 1 - \frac{2\sum_i p_i g_i}{\sum_i p_i + \sum_i g_i}$, where p_i is the prediction and g_i is the GT.

2.6.5 Bipartite Matching and the Hungarian Algorithm

Bipartite matching is used to uniquely assign a predicted box to a GT box [24], [52]. A common approach when employing set predictions by bipartite matching is to use the Hungarian algorithm to obtain the optimal assignments by minimizing the total cost. This cost is defined in a cost matrix, where each entry represents the cost of assigning a specific prediction to a GT object. The matrix can be constructed using a combination of different loss components depending on the task. By minimizing the total cost in this matrix, the Hungarian algorithm ensures a unique and optimal matching between predictions and GTs.

2.7 Principal Component Analysis

The PCA is a technique used to reduce high-dimensional data into a lower-dimensional space while retaining as much of the original variance as possible [53]. If there is a set of multi dimensional data, the first principal component is the direction along which the data points varies the most, the second is the direction along which they vary second most and so on. To reduce the dimensionality of a set from M to N one can therefore extract the N first principal components of that set of data and then project the data to the new coordinate system spanned by the principal components axes. Figure 2.5 illustrates a two dimensional example where the first and second principal component directions are shown.

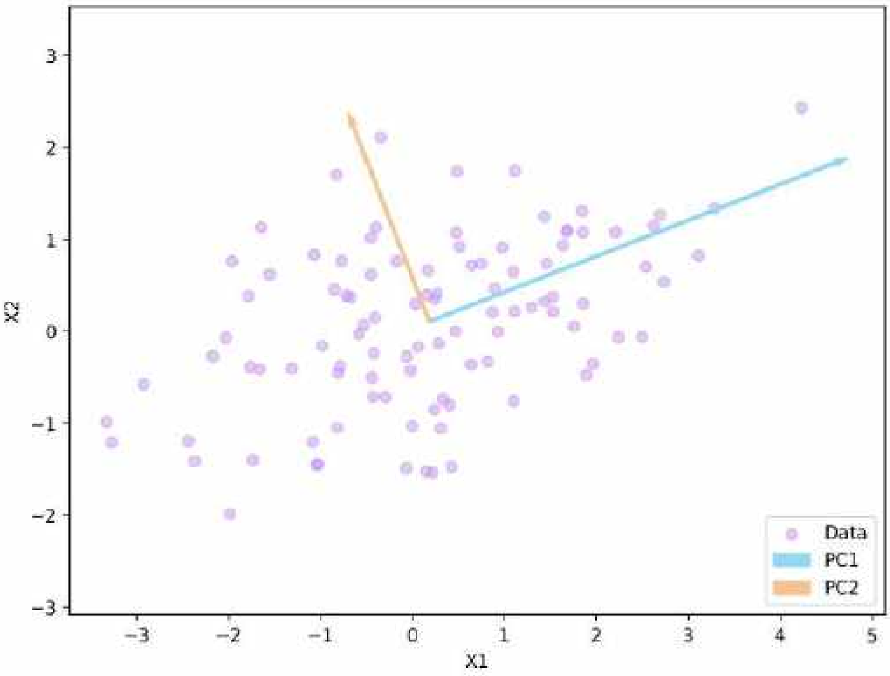


Figure 2.5: Illustration of how the principal components are calculated from a set of data.

3

Methods

This chapter presents a detailed overview of the methods employed during the project and highlights the design choices that shaped the approach.

3.1 Research Pipeline

This project aims to compare two self-supervised pre-training methods on LiDAR data by evaluating their downstream performance on 3D object detection. Two separate pipelines are developed, one using DINOv2 pre-training and the other using SAM2, as illustrated in Figure 3.1. Each pre-training strategy uses images and point clouds from ZOD to generate pseudo labels. Identical backbones composed of voxelization as described in Section 3.3.1, and a CNN are pre-trained using the generated pseudo labels as GTs. An object detection head can be attached to each pre-trained backbone to train and perform object detection from the learned features. In addition, a detector trained from scratch on LiDAR point clouds is included to serve as a baseline for comparison.

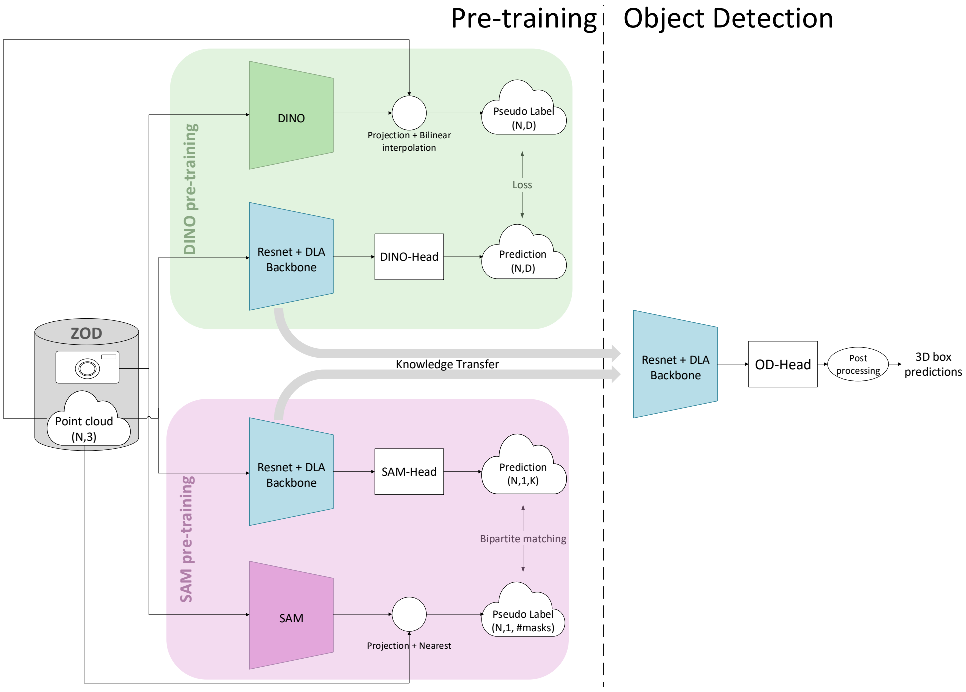
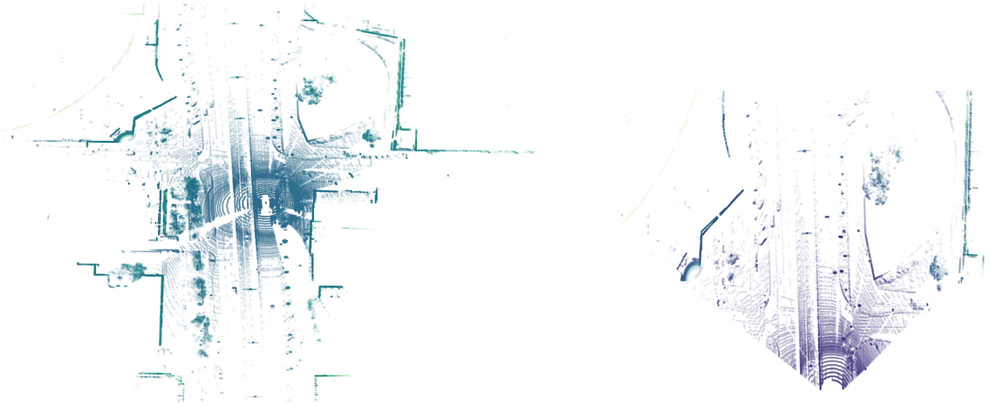


Figure 3.1: Full system diagram describing the pipeline to compare downstream performance of DINOv2 and SAM2 pre-training to fully supervised learning.

3.2 Data Processing

This project utilizes ZOD [54] which includes data from a calibrated camera-LiDAR setup with object detection annotations in the LiDAR point cloud. The camera images in the dataset portray the front view from the car and covers a range of 120°. However, the LiDAR point clouds in the dataset cover the entire range of 360° around the car in the BEV plane. To account for this and enable the usage of a calibrated camera-LiDAR setup, the LiDAR points outside the camera frustum are filtered out and all models are trained on the front 120° portion of LiDAR data. see Figure 3.2. It is vital to have the corresponding camera view to enable lifting image features via the foundation models and create pseudo labels.

To filter out points outside the camera frustum we simply transform all points in the raw LiDAR point cloud into the camera coordinate system using the extrinsic matrices of the camera and the LiDAR. All necessary calibration matrices for this procedure are pre-defined and can be found in the open dataset. The points which end up behind the camera (i.e. with a negative z-coordinate in the camera coordinate frame) after transformation are removed in this stage along points further away than 100 meters. The frustum is limited to 100 meters to avoid unnecessary complexity when defining the grid, as objects far away are not prioritised and points on them will be sparse due to the nature of the LiDAR sensor. The remaining points are projected into the camera image. Due to the fisheye characteristics of the camera in the dataset, Kannala projection [55] is used with the camera intrinsics and distortion vectors. Lastly, all projected points falling outside of the image are removed.



(a) Unprocessed point cloud covering 360°.

(b) Processed point cloud limited to camera frustum, covering 120°.

Figure 3.2: Pre-processing of the LiDAR point cloud to limit it to the camera frustum and a forward range of 100 meters in the vehicle's direction.

3.3 Object Detector

To enable the comparison of two pre-training methods, we implement an object detection baseline. This section describes the parts of the 3D LiDAR object detector which consists of a backbone and a detection head. The backbone is further split into an encoder from LiDAR points to feature map and a main CNN. The model is end-to-end differentiable for simple backpropogation in the training.

3.3.1 Feature Encoder

The LiDAR point cloud data is an unordered set of 3D points which is unsuitable to use directly as input to any network especially one based on convolutions which must be performed on dense images by construction. The backbone is made to remedy this and transform the raw data into a pseudo image which is the suitable input for a CNN.

This is done using the PointPillar voxalization procedure described in Section 2.1 due to its simplicity yet effective encoding of the positional relationship of the raw data points. However, in difference to the PointPillar paper [32] this project utilizes multiple layer feature encoding within each voxel to ensure that the resulting feature map captures the knowledge of the point distribution within that cell. The grid used to voxalize is defined in the BEV plane and the height dimension is unlimited within the pillars. We choose not to voxalize in the z-dimension as done in VoxelNet [30] since this would introduce additional engineering efforts to compress the feature tensors of a 3D grid into an image as well as computational complexity while improving the information captured very little since the points are not very spread out in z-direction. The voxels used in this project have width and depth equal to 0.125 m and the total BEV grid is limited from $[-128, 128]$ in x direction and $[0, 128]$ in y direction where the origin is placed at the centre of the LiDAR sensor.

The encoded feature vectors are then scattered into a pseudo image using the grid defined in the BEV plane. The resulting pseudo image will have the shape (H, W, C) where the Height (H) and Width (W) dimensions are the same as defined in the voxalization grid and the number of channels (C) are defined by the length of the encoded feature vectors chosen to be the same as in [32]. Due to the sparsity of LiDAR data, the resulting pseudo-image is padded with zeros where no points are present to create a dense image and a suitable input to a convolutional network.

3.3.2 Main CNN

The body of the detector network is based on the ResNet architecture, a widely used network within the deep learning community, proven effective in extracting valuable information from images. As the network was originally designed to operate on camera images, the first layers of the ResNet quickly compress the resolution of the input. Due to the sparsity of the LiDAR data the objects in the generated pseudo-image are very small. Therefore, this reduction in resolution may make the separation between objects difficult for the model. As a solution to this the first convolutional layers of the network are modified to have stride 1 and the pooling layer is removed which preserves resolution in the early stages of the detector.

As mentioned in Section 2.3, DLA is an effective way of combining rich semantic information from deeper layers with spatial information from shallower layers of the network. As mentioned above, objects in a LiDAR point cloud are often small, therefore it is beneficial to implement DLA from shallow layers, ensuring that spatial information is propagated through the network and integrated with information from the deeper layers.

The aggregation starts in the second residual block. Which has an output resolution only 4 times smaller than the original pseudo image, while incorporating information from the deeper layers with a larger receptive field. The resulting image of shape $(H/4, W/4, D)$, where D is the number of channels after the second residual block, is then propagated through the detection head to obtain the final predictions. The described architecture can be seen in Figure 3.3. Unlike the method described in [28], the convolution is applied to the deeper layer, after the upsampling. This choice was made to improve efficiency, as it saves computational resources in relation to the information it sacrifices. The computational savings outweigh the loss of information, which makes it beneficial for this project as the goal is not to optimize detection performance but to establish a baseline for comparisons.

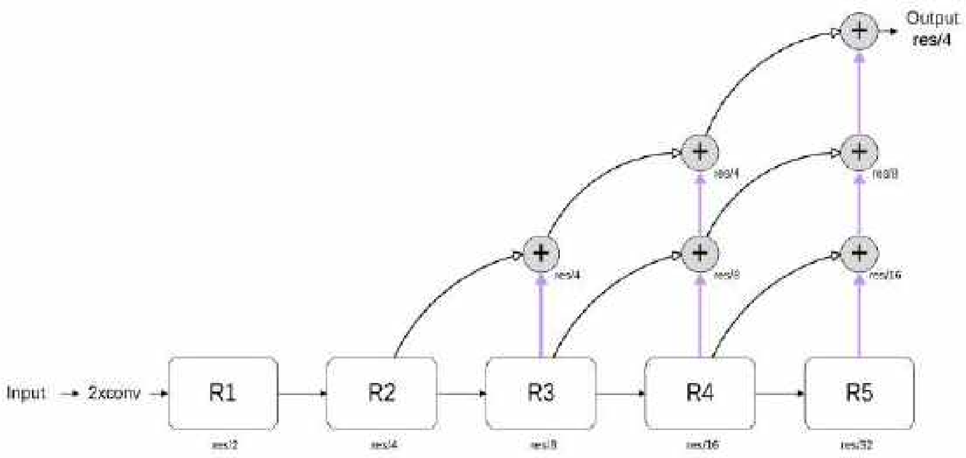
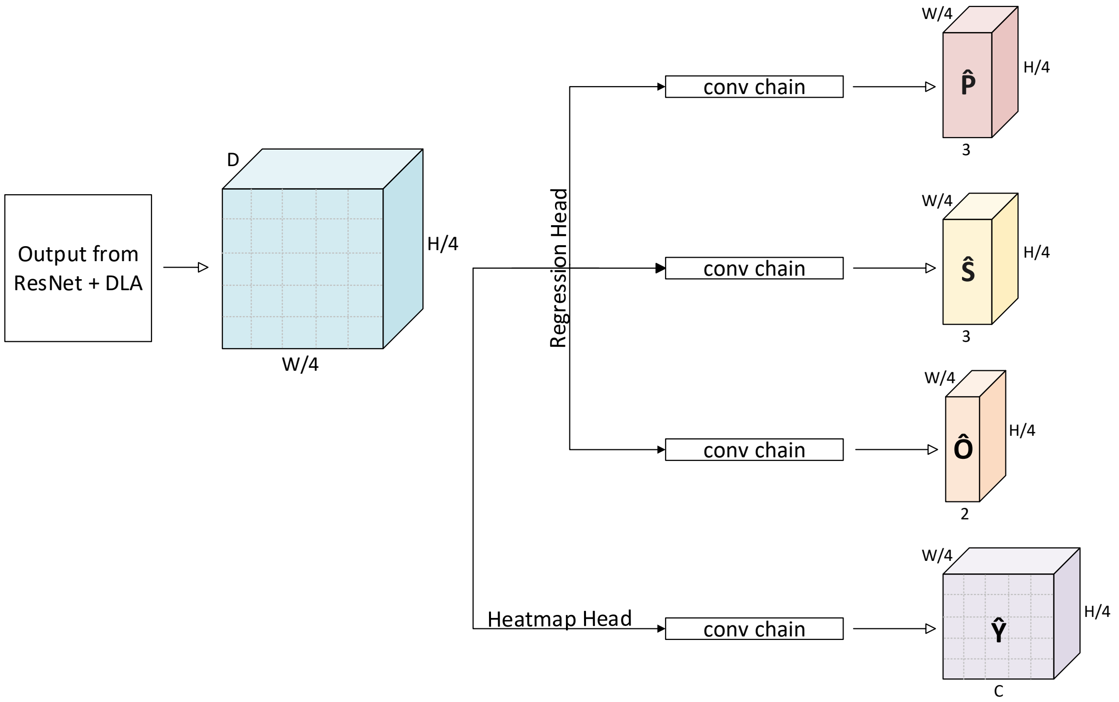


Figure 3.3: Structure of the DLA used in the model backbone. The aggregation begins in the second residual block and propagates through the network, leading to an output with dimensions downsampled by 4 compared to the input. Purple arrows represents interpolation followed by convolution. "Res" denotes the image resolution.

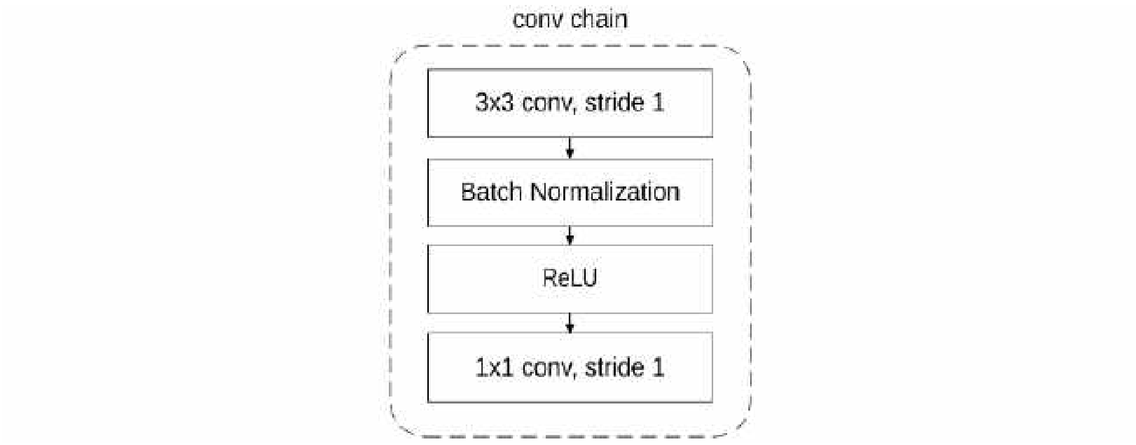
3.3.3 Detection Head

The object detector's head is largely inspired by the CenterNet architecture, as introduced in Section 2.4. It consists of two components, the regression head and the heatmap head. The regression head is modified in comparison to the CenterNet implementation to fit 3D point cloud data, as done in the CenterPoint framework presented in Section 2.5. The architecture used can be seen in Figure 3.4a. The regression head is responsible for predicting the bounding boxes, and is further divided into subheads that predict the position, orientation, and size of an object. These predictions are made by passing the output from the main CNN through a convolution chain illustrated in Figure 3.4b.

The heatmap head, on the other hand, generates a heatmap per class that represent the unnormalized posterior probability of an object of that class being present in each location of a grid. This head utilizes the same convolution chain as the regression head. The grid is defined in the BEV plane similar to the one used for voxalization but with a lower resolution, matching the resolution of the output of the main CNN. This means that one heatmap cell corresponds to a receptive field of 4×4 voxels in the voxalization grid.



(a) Architecture of the implemented heatmap head and regression heads. The regression head is further divided into sub heads that each predict different properties of the bounding boxes.



(b) The convolution chain used for the detectors head consists of a 3×3 convolutional layer followed by a batch normalization layer, a ReLU layer and finally a 1×1 convolutional layer.

Figure 3.4: Illustration of the object detection head architecture based on CenterNet [25] but adapted for 3D objects, including its convolutional processing chain.

The heatmap has shape $(W/4, H/4, C)$, where C is the number of classes. For each class, the heatmap contains ones where an object of that class is present in the receptive field corresponding to that heatmap location, and zero otherwise. The position is encoded as a $(W/4, H/4, 3)$ tensor including the $\Delta x, \Delta y$ from the grid cell center to the center of the object and the z coordinate at each location with an object present. Similarly, the size is a $(W/4, H/4, 3)$ tensor with the size from the center of the object to the edge of the box in each direction. The orientation is a $(W/4, H/4, 2)$ tensor with $\sin(\varphi)$ and $\cos(\varphi)$. φ denotes the yaw-angle which is the rotation around the z-axis. According to [33], this holds enough information to accurately represent the rotation of the boundary box.

3.3.4 Loss function

Similarly to [25] and [33], the annotations are expressed as a target heatmap, Y and target tensor for each of the regressed 3D object properties; position P, size S and orientation O. A Gaussian kernel is applied to the target heatmap as done in [25] to punish predicted peaks close to the true peak less than those far away.

The loss is calculated in two parts, a heatmap loss $\mathcal{L}_{HM}$ and a regression loss $\mathcal{L}_R$ weighted with $\lambda_{HM} = 1.0$ and $\lambda_R = 1.0$ such that the total loss is described as:

$$\mathcal{L} = \lambda_{HM}\mathcal{L}_{HM} + \lambda_R\mathcal{L}_R \tag{3.1}$$

The heatmap loss is a focal loss as given in Equation 2.1 and for the regression an L1-loss as described in Equation 2.3 is applied on the stacked regression tensors only on the locations where the target heatmap is one, i.e. where there exists a corresponding object. Figure 3.5 shows the target representations of the annotations and where each loss is applied.

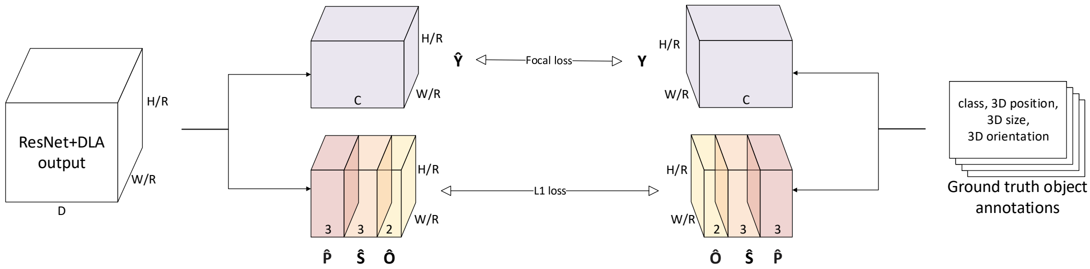


Figure 3.5: Illustration of how the loss function is applied. Providing a measurement of how similar the model output is to the generated targets. Here, the R stands for the reduction of resolution between the BEV grid used for voxalization and the model output. For the model described in this method is $R = 4$.

3.3.5 Data Augmentation

Data augmentation to the training data was done by rotating all point clouds and annotated boxes [56]. This method ensures that the model is exposed to different rotations, enhancing its ability to generalize across different orientations and making it less sensitive to spatial variates in a scene. This data augmentation technique was implemented by randomly selecting uniformly distributed rotation angles within a specified range $[-30°, 30°]$ and rotating the entire point cloud and corresponding annotations around the z-axis by the randomized angle. Despite this rotation, no points will be lost because the rotation angle is limited to 30 degrees. This restriction ensures that the rotated points remain within the valid range. In this implementation, the rotated point clouds and annotations replace the original data, meaning that the amount of training data was not increased but only altered.

3.3.6 Post-processing

During post-processing, predicted objects are only kept if their objectness exceeds a certain objectness threshold, eliminating unconfident predictions. The remaining predictions are then further processed to remove overlapping and redundant predicted objects.

In the CenterNet paper [25], the keypoint extraction made by maxpooling over the heatmap (which according to [25] replaces the need for a non-maximum-suppression) is applied per channel, i.e. per class. This means that for each class, only the object with the highest confidence score in a local neighbourhood is kept. This ensures that objects of the same class do not share the same grid cell. However, objects of different classes may have overlapping boxes, and they are treated separately in this approach.

Since the voxalization grid used in this project is defined in the BEV plane objects sharing bounding boxes can be seen as redundant as objects are rarely stacked vertically. Therefore, overlapping objects should be removed during post-processing, regardless of the class. This is implemented by incorporating Non-Maximum Suppression (NMS), which is not dependent on class, keeping only the object with the highest confidence score in cases where their bounding boxes overlap to a certain threshold.

3.4 Pre-Training with DINOv2

The DINOv2 model has strong capabilities of identifying object boundaries, classes and background in 2D [11]. To enable the distillation of these properties into the object detector model a pre-training pipeline which generates a DINOv2 feature per 3D point in the original point cloud is constructed. The backbone of the object detector is kept as the one described in 3.3.1 while a separate DINO-head is applied on top that generates a feature cloud and enables comparison between predicted and true DINOv2 features. The following sections describes the feature cloud labelling pipeline, the separate DINO-head and the loss function used in the pre-training.

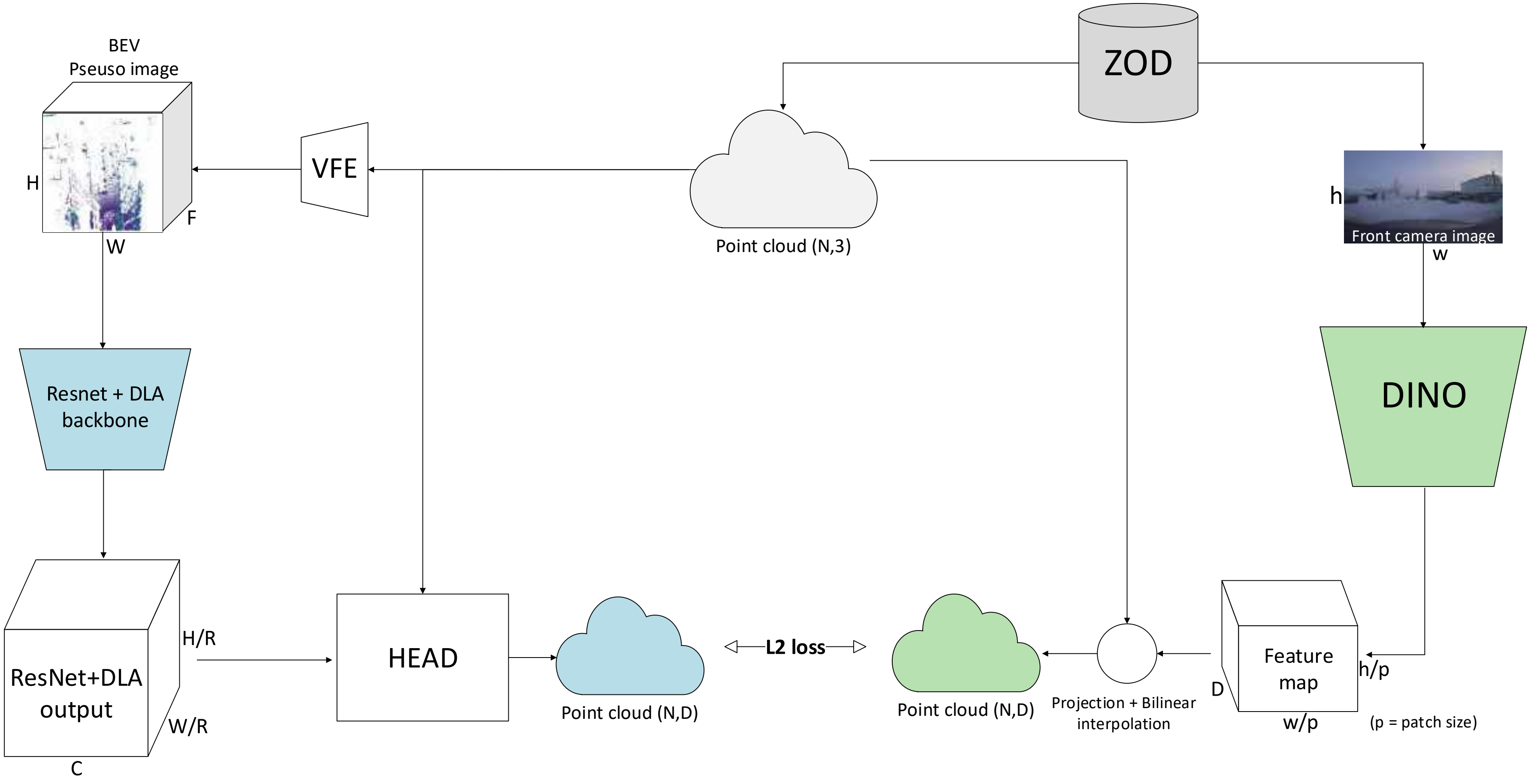


Figure 3.6: Illustration of the DINO-pretraining pipeline. Images are fed though the DINOv2 model after which the 3D points are projected onto its output and assigned a feature vector using bilinear interpolation. The point cloud is also fed into the ResNet with DLA backbone and through a DINO-head which produces a feature cloud that can be directly compared with the DINOv2 produced feature cloud using L2-loss.

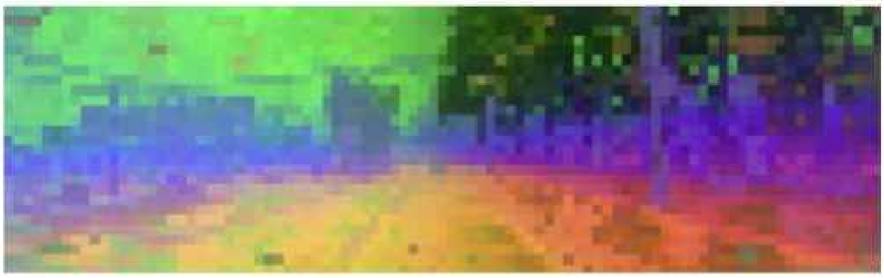
3.4.1 Pipeline

The pseudo-labelling pipeline lifts outputs of the DINOv2 model to 3D by projection and bilinear interpolation, this produces a "pseudo-label" in the form of a feature cloud of shape (N, D) where N is the number of points in the input cloud and D is the length of the channel dimension in the DINOv2 models output. The DINOv2 model used in this project is the vit_base_patch14_dinov2.lvd142m which provides an output channel size of $D = 768$ and a patch size $p = 14$.

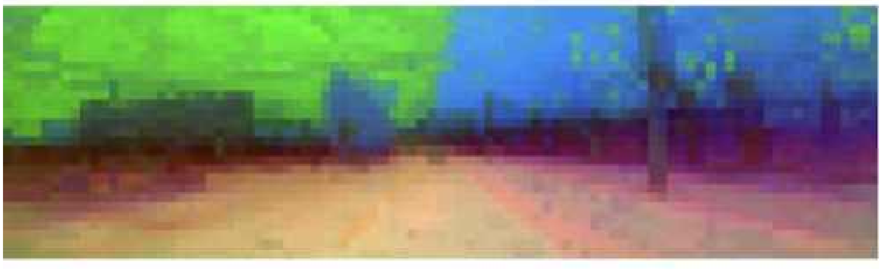
The ZOD dataset provides a calibrated setup of camera images and LiDAR point clouds. The images are fed into the model which outputs a feature grid of shape $(h/p, w/p, D)$ where $(h, w, 3)$ is the input image shape. The point clouds are filtered to keep only the points inside the camera frustum corresponding to the input image and the 3D points are then projected onto the DINOv2 output feature grid. Each projected point is assigned a feature vector of length D using bilinear interpolation to extract a combination of the feature vectors in the surrounding grid cells. The result is a feature cloud of shape (N, D) which can be used directly in the loss function.

3.4.1.1 Smoothing Positional Encoding

The original DINOv2 output feature grid includes sharp and abrupt changes (See Figure 3.7a). As a result of this the model may struggle to create a continuos BEV representation from which the predicted feature cloud is generated. In the paper "Denoising Vision Transformers (DVT)" [57] a denoising method is proposed which smooths the positional encoding in the DINOv2 feature map. Incorporating this into the DINOv2 pre-training pipeline, the DVT model is applied directly after the DINOv2 model producing a smoother feature grid see Figure 3.7b.



(a) DINOv2 raw feature grid output.



(b) DINOv2 feature grid output after DVT.

Figure 3.7: DINOv2 feature grid output before and after DVT is applied.

3.4.1.2 Reduction of Feature Dimension using PCA

The DINOv2 model outputs feature vectors of length $D = 768$, which makes training both time- and memory-intensive. Moreover, learning such high-dimensional representations is likely too complex for the model architecture developed in this project. As the DINOv2 model is trained on the entire web, one can hypothesize that its features include much more information that what is necessary in a road setting. For instance, features describing if an object has characteristics similar to scissors, t-shirt, shoe, pencil, etc. is seldom important when training a model to obtain high results in a driving setting. Therefore, it is crucial to extract the most informative features in order to enable effective learning and improve overall performance and tailor the extracted features to fit the specified task. To obtain this, a PCA is performed on a part of the training set (1000 samples) to extract the components in the DINOv2 features most meaningful in the self-driving context, i.e. most present in the ZOD dataset. The largest n components are extracted where $n = 32$ was found to give best downstream results on the most common class empirically see Table 4.1, and the PCA fitting is kept fixed throughout the training. The PCA is applied following the DVT and the number of output channels as well as the hidden dimensions in the DINO-head are changed to $D = n = 32$.

3.4.2 Feature Prediction Head

To enable the prediction of a feature cloud comparable to the one generated from the DINOv2 output, a separate head was created and shown in Figure 3.8 which after the pre-training is replaced with the object detection head. The head consists of a convolutional chain as described in Figure 3.4b which generates a BEV feature map of shape $(H/R, W/R, D)$ where $R = 4$ is the resolution downsampling in the ResNet with DLA backbone. The 3D points from the camera frustum filtered point cloud is projected on to the feature map and each point is assigned a feature vector as a combination of the vectors in surrounding grid cells through bilinear interpolation.

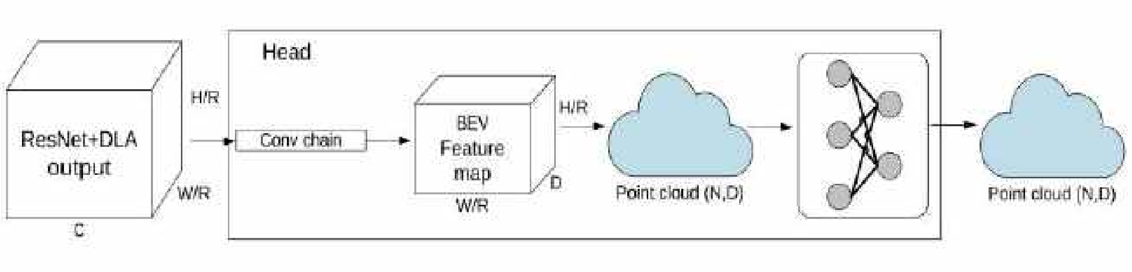


Figure 3.8: Illustration of the DINO-head. The ResNet + DLA backbone output is put through a convolution chain as shown in Figure 3.4b to generate a BEV representation of a DINOv2 feature map. The 3D points are projected onto the BEV map and assigned a feature vector through bilinear interpolation. The produces feature cloud is put through a three layer FFN to generate the final predicted feature cloud.

To retain as much information as possible a high channel dimension in the backbone output is desirable. However, the feature cloud is required to have a feature dimension of 32 to match the GT cloud. To bridge this gap, the predicted feature cloud is put through a FFN consisting of three fully connected linear layers with ReLU activation in between and hidden dimension D.

3.4.3 Removing Bleeding of DINOv2 Features

Due to the different mounting positions of the sensors (camera and LiDAR), points located behind objects may be projected onto the surfaces of those objects. This occurs because the LiDARs mounting position allows it to capture the object when the camera can not. Since the points are projected onto the frontmost pixel, they are assigned features corresponding to that pixel, rather than the pixel they would occupy based on their true position.

As object features leak onto the background behind the object, the model is provided with faulty information about the object properties present in the point cloud, this phenomena can be seen in Figure 3.9a. The bleeding issue was resolved by limiting points that can be assigned a feature by depth, hindering objects feature leakage onto the background and nearby surroundings. For every pixel, the depth of all points projected onto that pixel is calculated in the camera coordinate system. The depth is measured in reference to the point closest to the camera and only points under a certain "depth threshold" are kept. The maximum depth threshold is set to 3 meters, which means that one objects features can not "span" a depth greater that.

Since the DINOv2 output feature map is coarse with a resolution of only 91×27 pixels, a finer grid was constructed by dividing each pixel into 4 smaller pixels. This allows for a more detailed filtering of bleeding points leading to less information loss as fewer correctly assigned feature points are removed in the algorithm. The effects of the different pixel sizes used in the method is shown in Figure 3.10

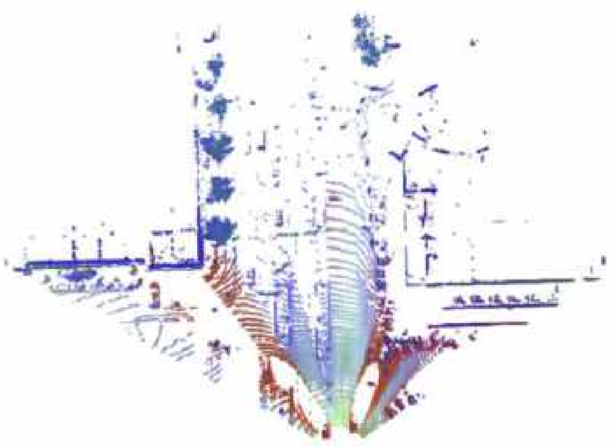
Comparing the feature clouds before and after implementing this method shows that the features are effectively bounded to their corresponding object, which can be seen in Figure 3.9. Moreover, the model trained only on non-bleeding points effectively learns to provide realistic feature predictions for these points. For example ground and vegetation features as can be seen in Figure 3.9c where the GT points were removed.

(a) Feature cloud with bleeding issue. Car features (in blue) are leaking behind the cars.

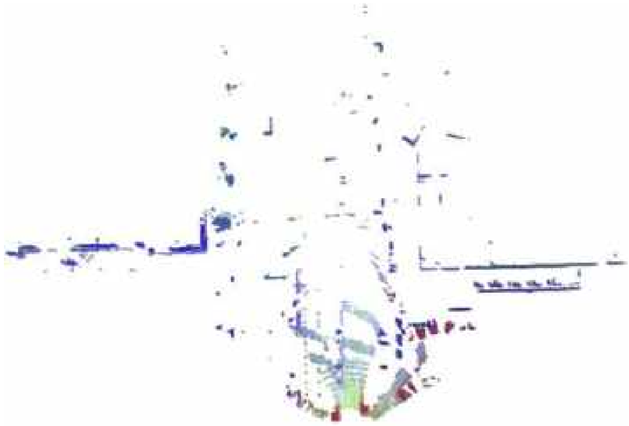
(b) Feature cloud after the bleeding points has been removed, car features are now limited in depth.

(c) Predicted feature cloud after the bleeding issue has been resolved, points behind the car are seen as background.

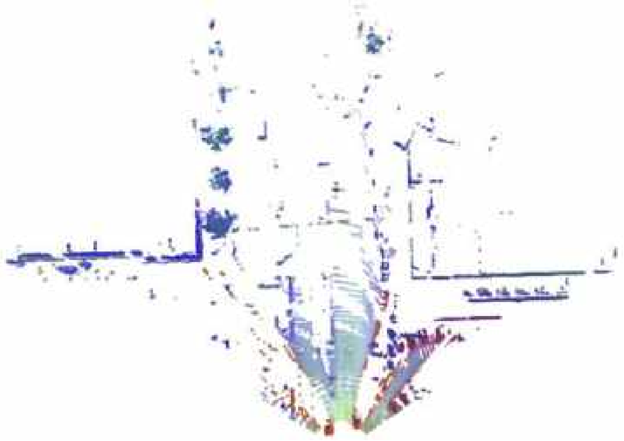
Figure 3.9: Comparison of the same feature cloud before and after implementing the bleeding issue fix. The PCA is consistent between images meaning that colours are comparable.



(a) Feature cloud with bleeding issue before removal.



(b) Feature cloud after the bleeding points has been removed using full pixels.



(c) Feature cloud after the bleeding points has been removed using 4 times the resolution of pixels.

Figure 3.10: Pixel size effect on bleeding removal algorithm. The Cloud after bleeding removal using full pixel has lost lots more information about the original cloud that that of the smaller pixel grid whilst also removing more bleeding points.

3.4.4 Object Detection Head

To reduce the gap between the object detection head and the head used during pre-training on DINOv2, a specific query based head was designed. The feature prediction head used for pre-training utilizes the extraction of backbone features to each point in the original point cloud. To to make it easier for the model to learn mapping from the pre-trained backbone features to objects during fine-tuning, a new head was designed. This head utilizes the same query based principle as the pre-training head. However, in this head the backbone output is queried at each grid cell centre in contrast to querying at point coordinates in the original point clouds. The features are extracted using projection and bilinear interpolation, which is then followed by a small FFN, see Figure 3.11. The produced feature cloud is then fed into two separate FFNs and scattered back to the original grid shape. This head outputs the same shape heatmap and regression output as the OD-head specified in subsection 3.3.3 making it straight forward to extract the final predicted bounding boxes and labels by using the same post-processing procedure as in subsection 3.3.6.

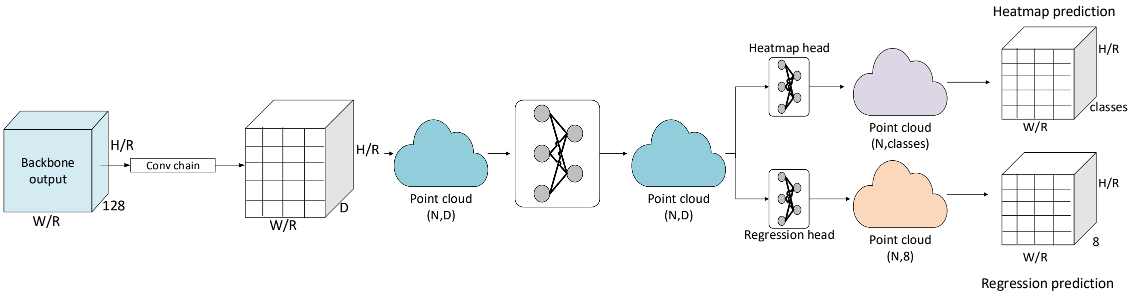


Figure 3.11: OD-Head designed specifically for DINOv2 pre-trained model.

3.4.5 Loss function

As mentioned, the DINO-head outputs a feature cloud of shape (N,D) which can be directly compared to the feature cloud generated using the DINOv2 model. The two clouds are compared point-wise using MSE-loss described in Equation 2.2. The loss is averaged over the number of points (N). This allows the model to concentrate on challenging samples, as the MSE-loss places greater emphasis on areas where the model's predictions are inaccurate.

SmoothL1loss, defined in Equation 2.4, was also tested for the task. However, through multiple trials and controlled comparisons MSEloss was found to be more suitable for the task.

3.4.6 Data Augmentation

LiDAR rays only capture the surface of objects, providing no direct information about their depth. To simulate depth of objects, noise is introduced to the query point cloud data that is to be projected onto the DINOv2 feature image. Specifically, each point in the point cloud is disturbed by extending it slightly along its normal vector simulating penetration into the object.

For this augmentation, each point is assigned a noise value Δ sampled from a uniform distribution in the range [0,0.1] meters. This range is chosen to approximate a plausible minimal thickness, reflecting the intuitive idea that object surfaces have depth. The point's position is then extended by displacing it by Δ along its normal vector, thereby pushing it into the object.

In addition, we add the same augmentations as for the object detection described in subsection 3.3.5 to the original point cloud that is fed into the model during training.

3.5 Pre-Training with SAM2

The SAM 2 model is a powerful segmentation model trained on large-scale web data, making it a suitable teacher for training an object detector [16]. It is specifically designed to generate object masks for any class based on spatial input prompts. In this project, we develop a pseudo-labelling engine that uses SAM 2 and specifically the sam2.1_hiera_base_plus version to generate 2D segmentation masks from camera images, which are then lifted into 3D, as illustrated in Figure 3.12. These 3D masks, or pseudo-labels, serve as ground truth for pre-training the object detector's backbone to predict 3D class-agnostic segmentation masks. To enable this process, we design a dedicated pipeline, shown in Figure 3.13.

3.5.1 Pseudo-Labelling Engine

The pseudo-labelling engine shown in Figure 3.12 is used to create 3D segmentation masks. These masks are then utilized as ground truth masks in the loss function as presented in subsection 3.5.3 for SAM pre-training. The engine is derived from the pseudo-labelling engine presented in [13] and can be divided up into four main parts. First, masks are obtained from SAM using input camera images. NMS is then applied to retain only distinct masks, which are subsequently lifted onto the 3D point cloud resulting in 3D binary pseudo masks.

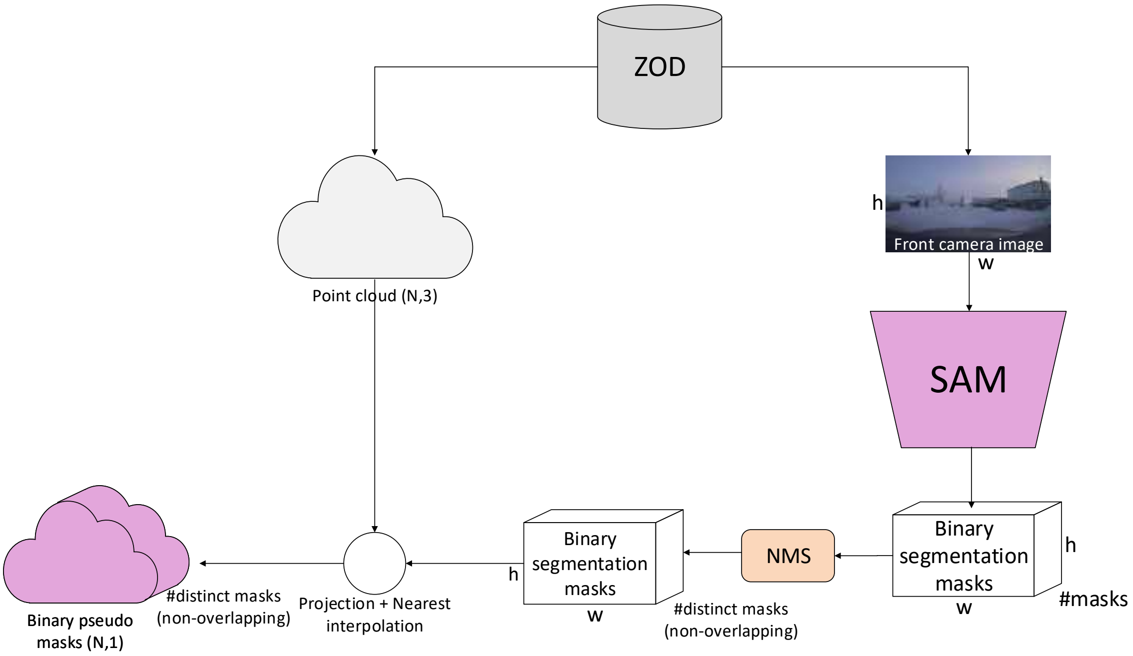


Figure 3.12: Pipeline of the SAM pseudo labelling engine. Images from ZOD are fed to the SAM 2 model, which outputs all predicted masks for each image. NMS is then performed based on the masks' areas, which suppresses masks that are part of another masked object, resulting in a tensor of shape (h, w, #distinct_masks). Through projection and bipolar interpolation, a resulting binary pseudo mask tensor of size (N, 1, #distinct_masks) is created.

Following [13] the parameters listed in Table 3.1 were used when obtaining the masks from SAM on the input image. Instead of prompting SAM with points or text, which are both common approaches, the model is promoted with a grid that corresponds to the input image, but with a lowered resolution (points per side). This allows the model to generate masks that indicate which region or regions each cell belongs to. The number of points per side was set to 16 which gives a good trade-off between resolution versus computation time. In contrast to [13], the threshold for minimum mask region was 25 instead of 100. The reason behind using a smaller value is to keep masks of objects such as poles or traffic signs, as they are often small in size.

Points Per Batch	Predicted IoU Thresh	Stability Score Thresh
64	0.84	0.86

Table 3.1: Parameter values based on [13].

To retain only distinct masks, NMS is applied to the binary masks produced by SAM. The NMS is performed such that area is prioritized over confidence score, in order to favour complete objects over smaller parts such as tires, windows, and similar components, as proposed in [13]. The points in the original point cloud are then projected onto the selected distinct masks and each point is assigned the nearest binary value of it's projection. This results in a set of distinct binary 3D masks.

3.5.2 Pre-Training Pipeline

The pipeline of the SAM2 pre-training is visualized in Figure 3.13. A LiDAR point cloud is fed into the backbone of the object detector described in Section 3.3.1. The backbone output feature map is then put trough a SAM-head which consists of a transformer part and a feature cloud generation part. Finally the output of the SAM-head is compared to the ground truth pseudo-labels in the loss function.

The SAM-head consists of two main parts. First the points in the original cloud are projected onto the backbone output to extract a feature cloud which is further processed in a three layer MLP similarly to the DINO-head in Figure 3.8. Next, the backbone output and a set of learnable queries is put through a Transformer consisting of 6 layers of self- and cross-attention with four heads. The transformer is added to enable prediction of distinct masks corresponding to different queries. Lastly, the feature cloud and output queries are combined through matrix multiplication to produce the final predicted masks for each query.

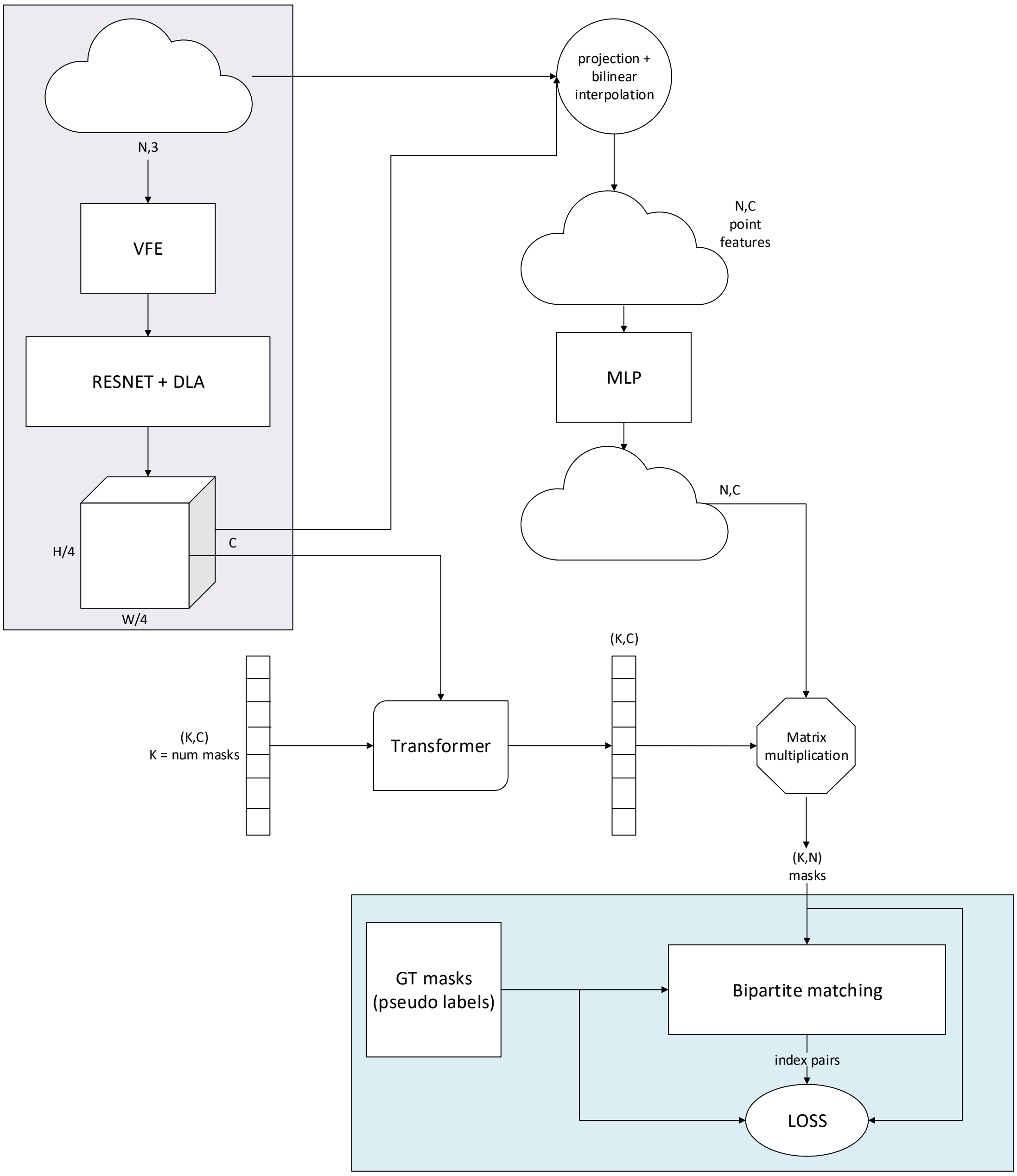


Figure 3.13: Illustration of SAM pre-training pipeline. Marked in lilac is the backbone from subsection 3.3.1. Marked in blue is the loss function. The SAM-head generates (K,N) binary masks from the backbone output.

3.5.3 Loss function

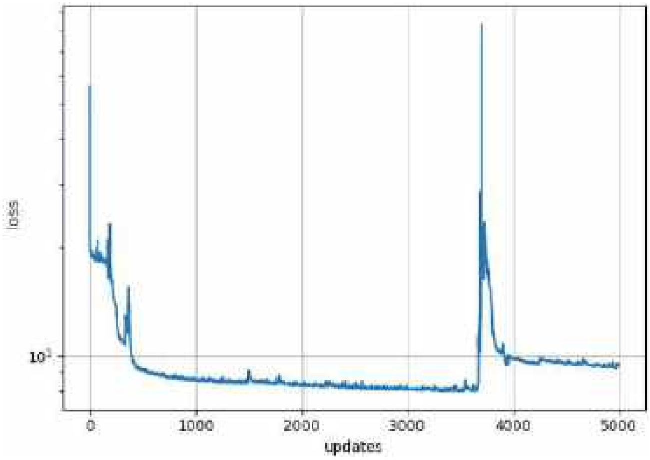
The loss function in the SAM-head is inspired by the paper [58] which uses mask prediction for panoptic segmentation. Similarly to [58], the SAM-head uses bipartite matching to find the pairs of predicted masks ($\hat{m}$) and pseudo ground true masks (m) which have the highest similarity or lowest cost using the Hungarian algorithm. The cost matrix used in the Hungarian algorithm is calculated by summing the dice loss with an element-wise probability loss for each pair of masks as $\mathcal{L}_{mask} = \lambda_{dice}\text{Dice}(m_i, \hat{m}_j) + \lambda_{prob}\sum_{p=0}^{N}\frac{|\hat{m}_i^p - m_j^p|}{N}$, where N is the number of points in the point cloud, $\lambda_{prob} = 1$ and $\lambda_{dice} = 1$. This differs from [58] which uses a combination of Dice and BCE loss for the masks summed with the probability loss of that mask being of a specific class. As the pre-training problem in this project does not require the prediction of a class this part is left out.

The usage of Dice cost combined with a point-wise probability loss in the cost matrix is selected by empirically testing three alternatives. The alternatives tested in the cost matrix are Dice with BCE, Dice only and Dice with probability loss. Figure Figure 3.14 shows the loss calculated from the matched pairs over updates when applying the three different cost calculations in the bipartite matching. As the Dice with point-wise probability loss Figure 3.14c provides a fast convergence while having the smoothest behaviour this was deemed the most robust solution and is used in all further experiments and trainings.

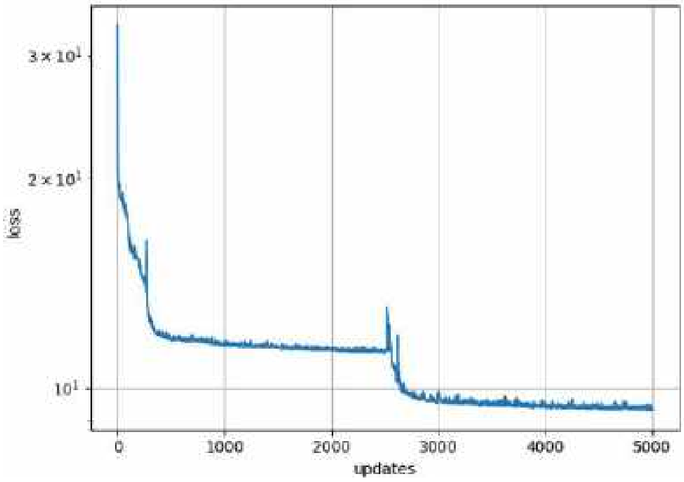
The loss function used for back-propagating and updating model weights is the combined Dice loss with BCE calculated on the matched pairs from the Hungarian algorithm. If K is the number of GT masks (and therefore also the number of matched masks) and $\hat{\mathbf{m}}_{\neg K}$ is the set of predicted but unmatched masks, then the loss function is given as:

$$\mathcal{L} = \lambda_{bg}BCE(\mathbf{0}, \hat{\mathbf{m}}_{\neg K}) + \sum_{i=0}^{K}(\lambda_{dice}\text{Dice}(m_i, \hat{m}_i) + \lambda_{bce}\text{BCE}(m_i, \hat{m}_i)) \tag{3.2}$$

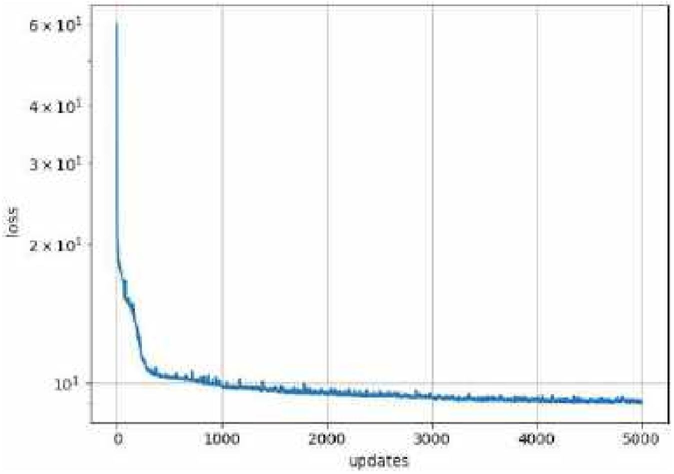
where $\lambda_{bce} = 1.0$, $\lambda_{bg} = 0.1$ and $\mathbf{0}$ is a set of zero masks the same size as the set of unmatched masks used to force the model to predict background masks. Due to the class imbalance between background and foreground ($K << \neg K$) the background loss is down weighted by 0.1.



(a) Dice and BCE loss. Model learns fast but has erratic behaviour.



(b) Cost based on Dice only. Model plateaus at high loss early before getting out of the local minimum.



(c) Dice and point-wise probability loss. Model learns quite fast and has smooth learning curve.

Figure 3.14: Comparison of losses used in cost matrix for bipartite matching. The losses were compared when overfitting the model to ten samples.

3.6 Data Augmentation

To avoid a domain gap between pre-training and fine-tuning in both pre-training procedures, the same augmentation as for the object detection described in Section 3.3.5 is applied to the original point cloud that is fed into the model during training.

3.7 Evaluation of Object Detection Performance

When evaluating the object detector, the Mean Average Precision (mAP) metric is used to gain understanding about the overall performance. This metric is calculated based on Precision and Recall which in turn is calculated from the number of True Positives (TP), False Positives (FP) and False Negatives (FN) as seen in Equation 3.3 and 3.4. The predictions are labelled as TP if the Intersection over Union (IoU) between GT and predictions is sufficient and FP otherwise.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.4}$$

IoU is a measurement of the overlap between a ground truth and a predicted box, reflecting to which extent the prediction is correct. Using a low IoU threshold will label instances that only overlaps a little with a GT bounding box as TP while a larger IoU threshold will cause the same prediction to be labelled as FP and thus may increase the amount of FN if a ground truth box does not overlap enough with any predictions. Therefore, different IoU thresholds (see Table 3.2) are used when determining a match between a ground truth and predicted box and averaged over. The calculations of IoU between boxes are simplified to exclude rotations, as the 2D boxes in the BEV-plane are calculated using Axis-Aligned Bounding Boxes (AABB).

Another threshold that largely effects the number of TP, FP and FN is the objectness threshold described in Section 3.3.6. A low objectness threshold causes the output to include too many boxes and therefore results in a high FP while a high objectness threshold has the opposite effect and results in a higher FN. To analyze the trade-off between recall and precision, the objectness threshold is varied to generate a precision-recall curve where the objectness values are shown in Table 3.2.

The precision-recall curve illustrates the trade-off between false positives and false negatives as recall indicates the model's ability to correctly identify all positive instances, while precision reflects its accuracy when making positive predictions. A higher recall indicates a tendency to favour identifying as many relevant instances as possible, possibly at the cost of introducing more false positives. On the other hand, higher precision means accurately making positive predictions, but with a greater focus on avoiding false positives, potentially increasing false negatives. By calculating the area under the precision-recall curve the mAP is obtained, where mAP $\in [0, 1]$. The mAP provides a balanced evaluation method, considering both precision and recall when evaluating the models performance.

In general the mAP is the area under the precision-recall curve when the recall spans the full range from 0 to 1. Having a recall of 1 would mean that the model finds all ground truths, as evident from Equation 3.4. Since object detection is a complex task, a recall of 1 is highly unlikely for any IoU and objectness threshold. Adding a point [0,1] in the precision-recall curve enforces a realistic boundary condition at the high-recall end of the curve. As a high recall would cause many false positives to be introduced, meaning that the precision must decrease following Equation 3.3. To extend the curve to cases where the recall is 0, the precision value at the lowest recall was extended to that point. This creates a limited area under the graph that is calculated using the trapezoidal rule.

In this project, the mAP is averaged over objectness thresholds and IoU thresholds. In addition, a total mAP is calculated, which is averaged over classes. In following sections mAP refers to the total mAP if nothing else is stated.

IoU :	[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5]
Objectness:	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]

Table 3.2: IoU and objectness thresholds used to calculate mAP.

3.8 Scaling Study

A scaling study is conducted to investigate the theoretical performance of pre-training DINOv2 when scaling to datasets larger than the one used in this project. To investigate this, pre-trainings are performed for dataset sizes 100, 500, 1000, 5000, 10000, and 89972 samples, where the last value is the full available training-set in ZOD. The validation of each model uses the full 10023 sized validation set. Each pre-training run until the validation loss converges, and the best model is selected based on the lowest validation loss observed. A suitable curve is then fitted to these lowest validation losses to allow extrapolation of the validation loss achievable when pre-training on even larger datasets.

For each pre-training setup, fine-tuning is performed until the vehicle mAP converges, enabling evaluation of the effect of pre-training sample size on downstream performance. Each pre-trained model is fine-tuned using 10 and 100 data samples, with results averaged over 10 runs. This setup is designed to specifically assess performance in few-shot detection scenarios.

4

Results

This chapter provides a description of the experiments conducted and presents the results of this thesis.

4.1 Feature Size Reduction Analysis with PCA

To investigate which DINOv2 feature length that gave the best result during pre-training the following analisys was conducted. The original feature dimension $D = 768$ was reduced using PCA to length 8, 16 and 32. The model was then trained to predict DINOv2 features of this length for each experiment and results are shown in Table 4.1. Epochs refers to the number of epochs the model took to pre-train until convergence and OD performance refers to the object detection performance when fine-tuned on 100 samples until validation loss converges, the mAP is averaged over 10 runs.

# Features	# Epochs	Vehicle mAP	Pedestrian mAP
8	11	0.687	0.094
16	12	0.709	0.127
32	14	0.723	0.116

Table 4.1: PCA feature analysis results.

From Table 4.1 it is noticed that using 32 PCA features results in a slightly improved object detection performance on the vehicle class. As it is desirable that the pre-training method should scale to larger dataset beyond ZOD, this larger feature dimension of 32 is used to capture more information. Therefore, following results concerning DINOv2 pre-training has been carried out using this feature length.

4.2 Pre-training validation performance

The DINOv2 and SAM 2 pre-trainings are ran on Nvidia A100 GPUs. To evaluate how well the models have learned the pre-training tasks before fine-tuning the validation set was used to generate visual representations of the pseudo-label and prediction.

4.2.1 DINOv2

The model was trained on the entire ZOD training set (89972 samples) with the parameters shown in Table 4.2. The training took two days and six hours until the best validation loss was reached.

Learning Rate	0.0003
Batch Size	4
# GPUs	1
# Epochs	14

Table 4.2: DINOv2 pre-training settings.

The visualisations in Figure 4.3, column one and two, show that the model has effectively learned to generate distinct DINOv2 features for objects based on a LiDAR point cloud. The same PCA reduction is used for generating RGB values from the features meaning that the colours are comparable.

4.2.2 SAM

The model was trained on the entire ZOD training set (89972 samples) with the parameters shown in Table 4.3. The training took nine days and 13 hours until the best validation loss was reached.

Learning Rate	0.0003
Batch Size	4
# GPUs	4
# Epochs	35

Table 4.3: SAM pre-training settings.

The visualisations in Figure 4.1 show that the model has learned to generate binary segmentation masks from a LiDAR point cloud corresponding to the pseudo labels. The masks are sorted such that a predicted mask corresponds to the GT mask of the same colour for each sample.

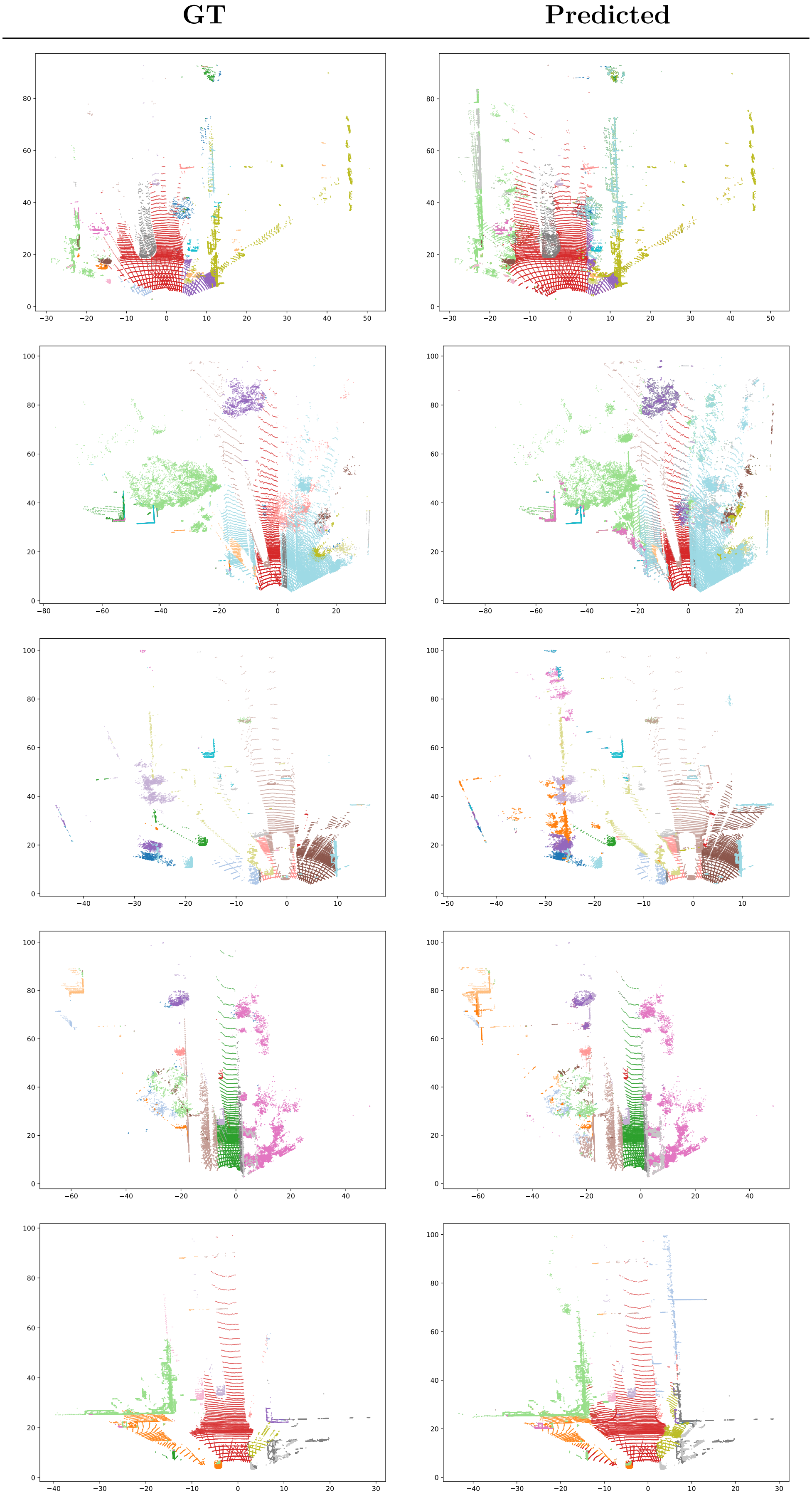


Figure 4.1: Comparison of GT and predicted masks on 5 validation samples for SAM pre-training.

4.3 Ablation Study DINOv2 pre-training

This thesis purposes a solution to the feature bleeding issue present in the DINOv2 pre-training. To investigate the effectiveness of this heuristic solution a small ablation study has been conducted. The DINOv2 pre-training with and without bleeding fix implemented ran until validation loss converged on the entire dataset. Figure 4.2 shows the validation losses for both models until the epoch where the best validation loss was reached. Both models were then fine-tuned on 100 samples on object detection and the results are shown in Table 4.4. Visual representation of and predicted clouds with and without bleeding fix enabled compared to the ground truth clouds are shown in Figure 4.3

Bleedfix	Best val loss*	Vechicle mAP
✓	138.21	0.723
	149.64	0.741

Table 4.4: Ablation analysis on effects of the bleeding fix algorithm. *Val loss refers to the pre-training validation loss.

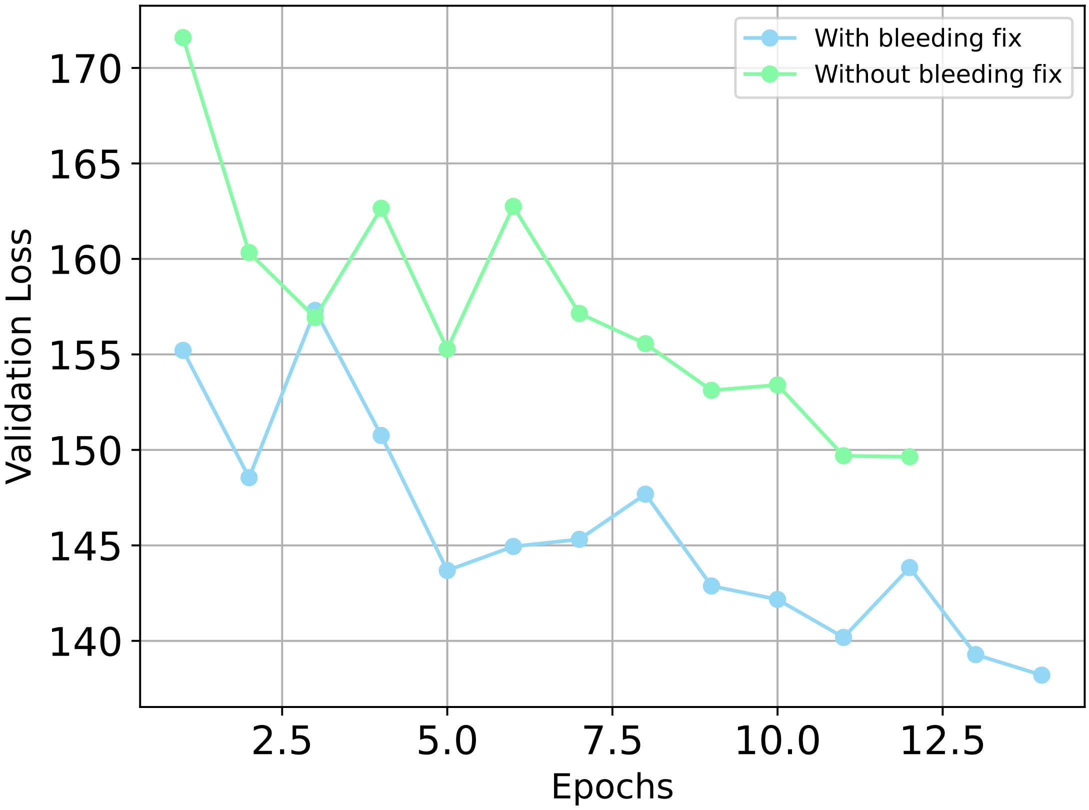


Figure 4.2: Comparison of validation losses with and without the bleeding fix.

4.4 Object Detection Performance

To evaluate and compare the different pre-training strategies effectiveness on object detection against each other and no pre-training, the models are fine-tuned on several different numbers of samples from ZOD with 3D object annotations. For all pre-trained models, the backbone is first kept frozen until validation loss convergence (evaluated on the full validation set). This allows for a measurement of how relevant the information in the backbone output feature map is when used in the context of object detection. The training is then continued for both backbone and the head together, using the same learning rate, until validation loss converges again. This allows for the evaluation of how well the full network adapts to the new tasks. The object detection performance on vehicles are shown in Figure 4.5. We focus on vehicle performance because, when fine-tuning on a single random sample from the dataset, vehicles are the only class with a high likelihood of being present in that sample.

FSOD	From Scratch Object Detection
PT-DINO	Pre-Trained on DINOv2 feature clouds
PT-SAM	Pre-Trained on SAM 2 pseudo labels

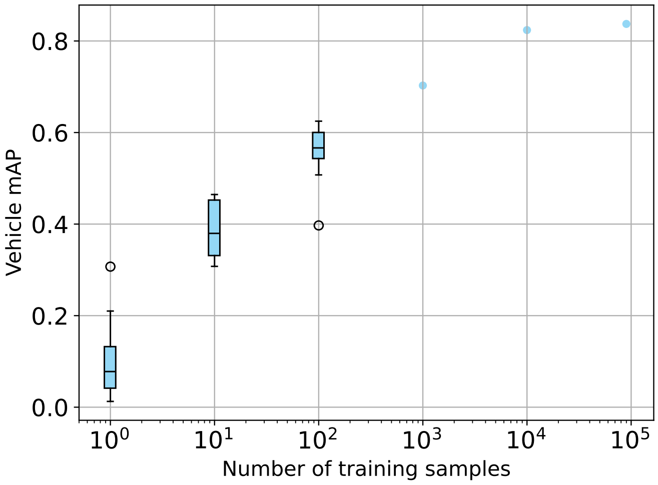
Table 4.5: Legend label definitions used in the plots.

Learning Rate	0.0003
Batch Size	6
# GPUs	1

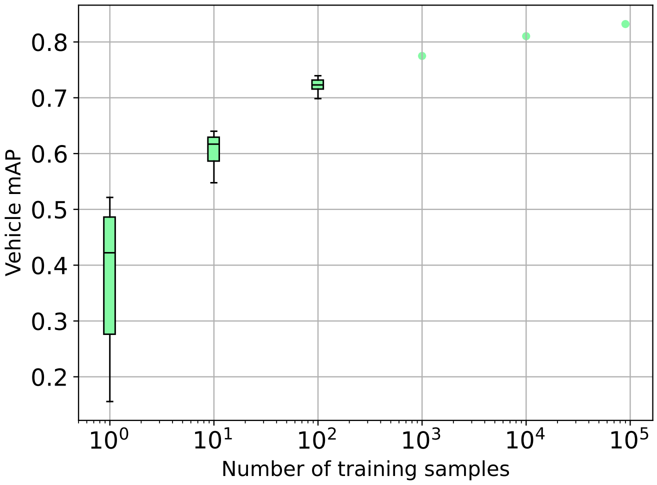
Table 4.6: Fine-tunings settings.

Each model, named as given in Table 4.5, is fine-tuned on varying sizes of training samples annotated for object detection and using the training settings reported in Table 4.6. For the smaller dataset sizes (ranging from 1 to 100 samples) each fine-tuning is repeated 10 times with different sets of samples. In a few runs, the mAP remained at zero throughout the entire training process. As this indicates abnormal behaviour and suggests that the model got stuck in a local optimum, we exclude these collapsed runs from our results. The number of such excluded runs for each fine-tuning configuration is reported in the "Collapsed" column of Table 4.7.

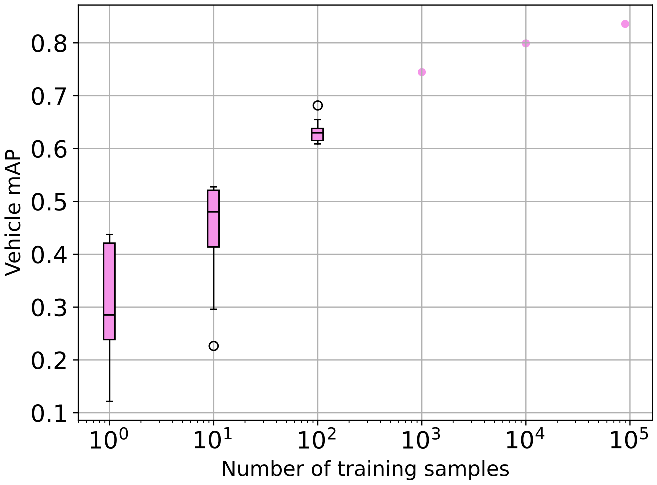
To identify additional outliers, the remaining measurements are shown in Figure 4.4, where outliers are marked with circles. Outliers are defined as values that deviate by more than 1.5 times the Inter Quartile Range (IQR) from the box. Before generating Figure 4.5, these outlier measurements are removed. The number of excluded values due to being outliers is reported in the "Outliers" column of Table 4.7.



(a) Identifying outliers in FSOD training.

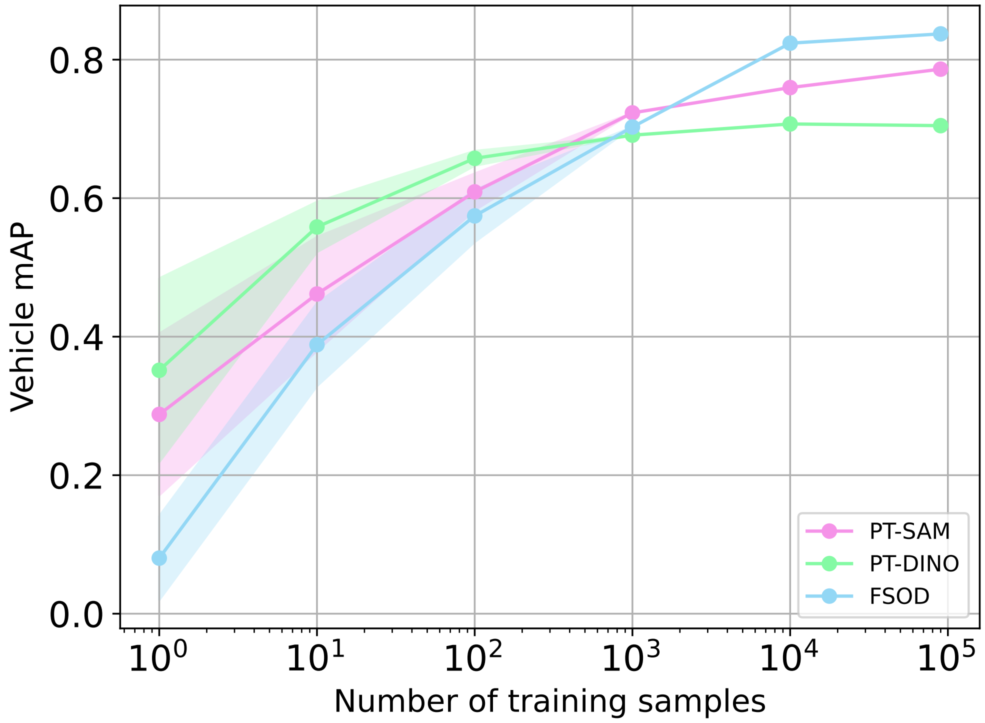


(b) Identifying outliers in DINOv2 fine-tuning.

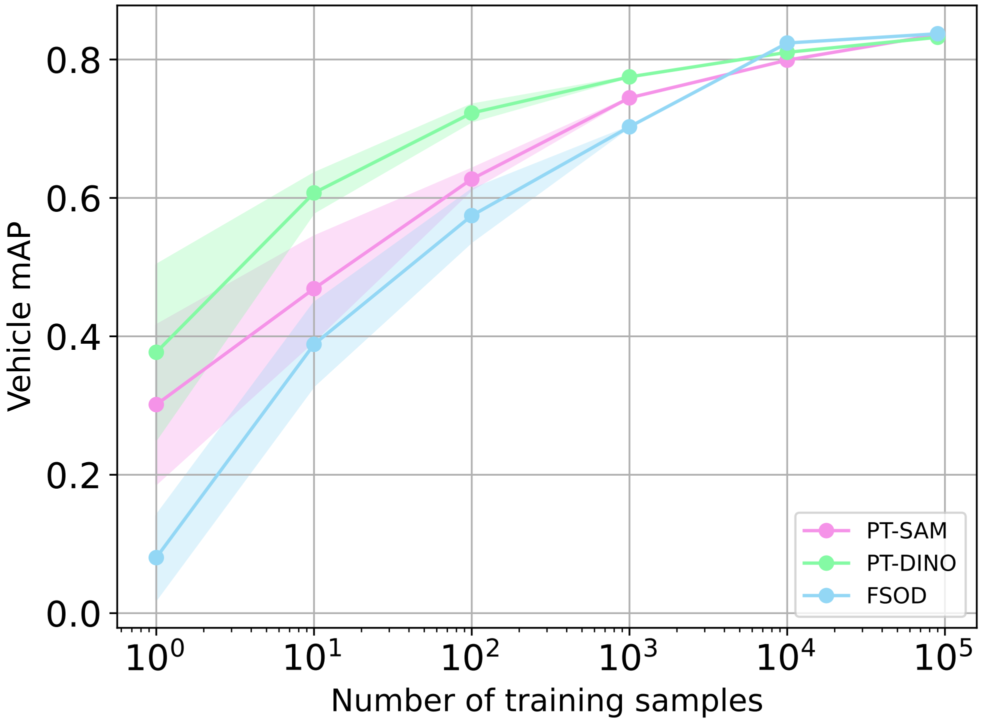


(c) Identifying outliers in SAM 2 fine-tuning.

Figure 4.4: Boxplots of Vehicle mAP over number of fine-tuning samples.



(a) Vehicles frozen🧊



(b) Vehicle unfrozen🔥

Figure 4.5: mAP for the different pre-training methods over the number of annotated data samples used for training the model.

Model	Samples	Collapsed	Outliers
FSOD	1	2	1
	10	0	0
	100	1	1
DINO	1	0	0
	10	0	0
	100	0	0
SAM	1	1	0
	10	2	1
	100	1	1

Table 4.7: Number of runs removed due to mAP remaining zero during entire training or outlier for each model when stopping training based on validation loss convergence.

4.5 Scaling Study

Results of the Scaling Study that investigates how the number of samples during DINOv2 pre-training effects the pre-training validation loss and the object detection performance when fine-tuned on 10 and 100 training samples. The results also presents how the pre-training likely to improve when pre-trained on increased number of samples beyond the availability on ZOD by fitting a line to the experiment results.

Presented in Table 4.8 is the combined results of pre-training and fine-tuning performance, including predicted values for the validation loss in pre-training when scaling the training-data to larger datasets. Fine-tuning has been ran until vehicle mAP converges, and fine-tunings are averaged over 10 runs. "Samples" refers to samples in pre-training, while (10) and (100) is the number fine-tuning samples. Figure 4.6 visualises the trend of validation losses for different amounts of samples in DINOv2 pre-training.

Samples	**Epochs**	**Val loss**	**FT-Val loss (10)**	**Vehicle mAP (10)**	**FT-Val loss (100)**	**Vehicle mAP (100)**
100	126	217.39	5.147	0.412	3.887	0.597
500	129	180.68	5.928	0.519	3.388	0.549
1000	91	171.06	7.309	0.574	3.246	0.616
5000	95	150.31	5.498	0.554	3.240	0.684
10000	63	147.83	6.653	0.530	3.049	0.671
50000	12	140.66	5.690	0.610	3.107	0.655
89972	14	138.21	3.182	0.641	2.560	0.741
89972*		137.83				
500000*		134.13				
1000000*		133.10				

Table 4.8: Scaling Study of DINOv2 pre-training where * denotes predicted data points. Fine-tuning validation loss and mAP is averaged over 10 runs. The value in parenthesis represents the number of fine-tuning samples.

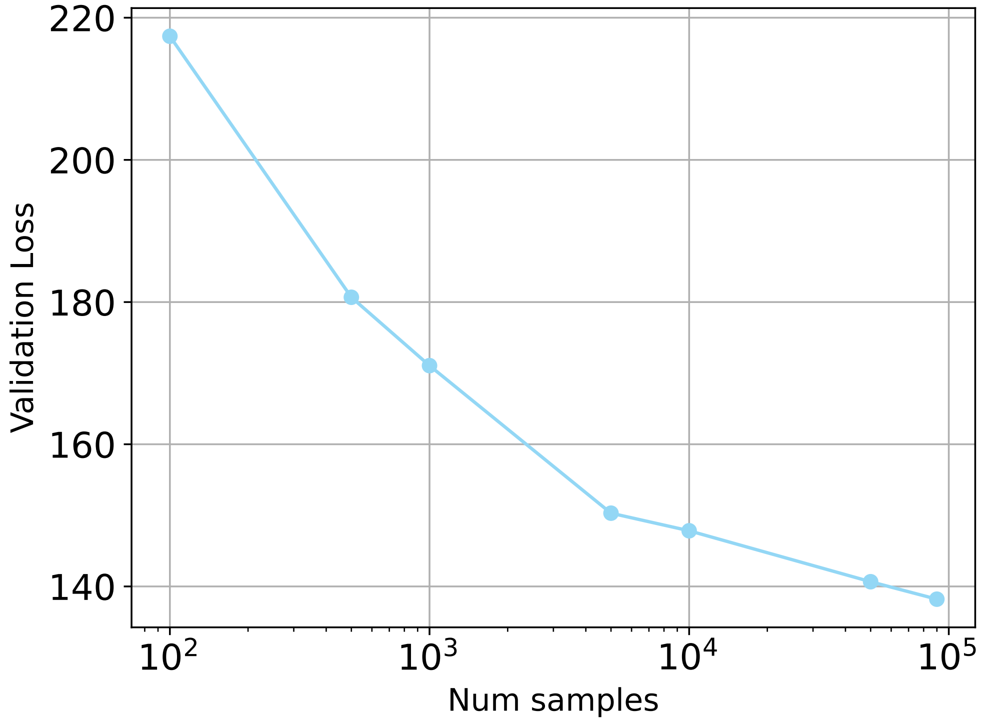


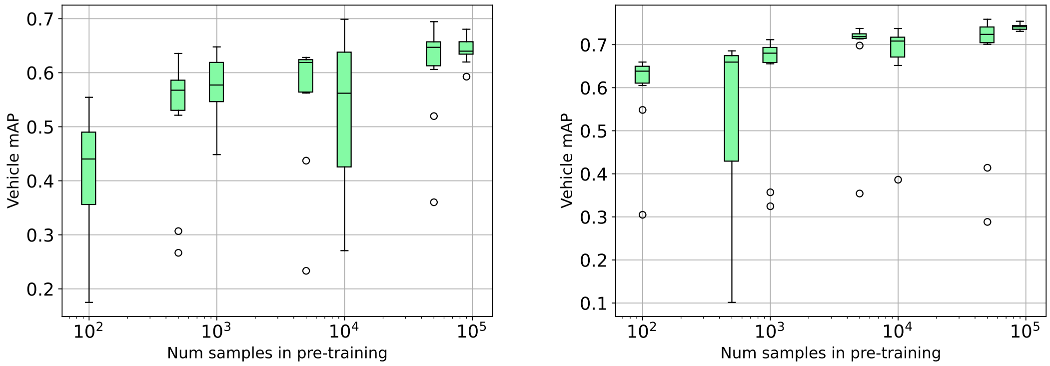
Figure 4.6: The best achieved validation loss over number of epochs used in DINOv2 pre-training.

To predict how the pre-training is likely to improve with a larger dataset a function was fitted to the observed values. The function used to fit the model to the observed data points is a inverted power-law shown in Equation 4.1

$$y = \frac{a}{x^b} + c \tag{4.1}$$

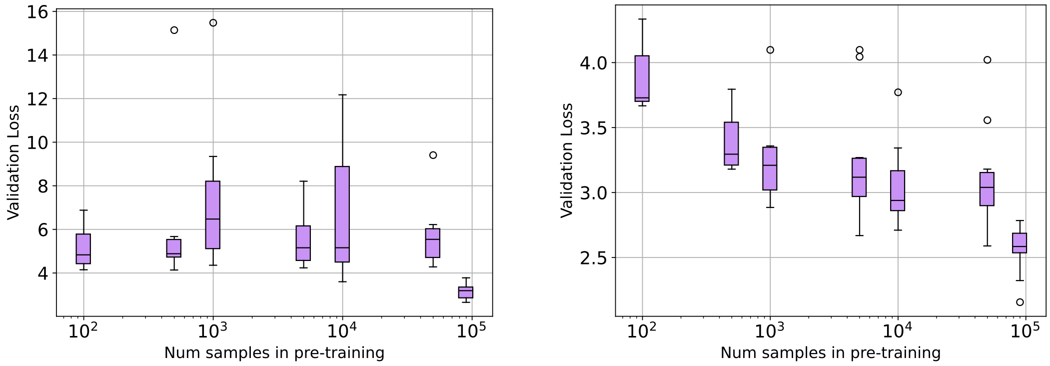
In Figure 4.7a, the last value among the obtained has been omitted when fitting the curve, then that value has been predicted. The predicted value for 89972 is 137.83 as seen in Figure 4.7a, whereas the real value presented in Table 4.8 is 138.21. Comparing the predicted value with the obtained value for the same data point in Figure 4.7b verifies that the fit is suitable. After this, the curve is fitted to all observed values resulting in final predicted values for 500K and 1M samples.

The downstream performance after pre-training DINOv2 on different dataset sizes is shown in Figure 4.8. All observed values have been included when producing the plots, and outliers are defined as in Section 4.4.



(a) Vechicle mAP when fine-tuned on 10 samples over the number of samples in pre-training.

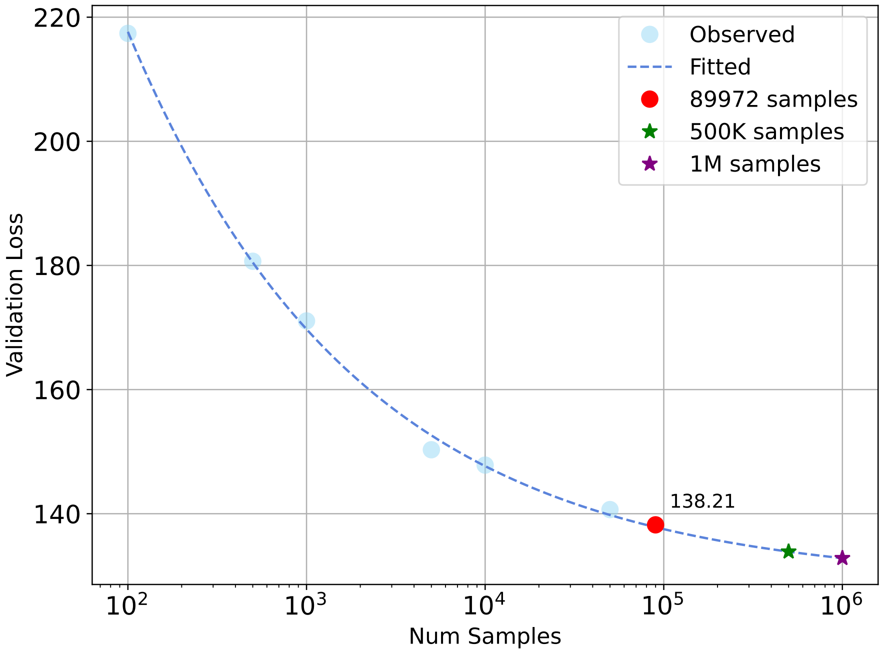
(b) Vechicle mAP when fine-tuned on 100 samples over the number of samples in pre-training.



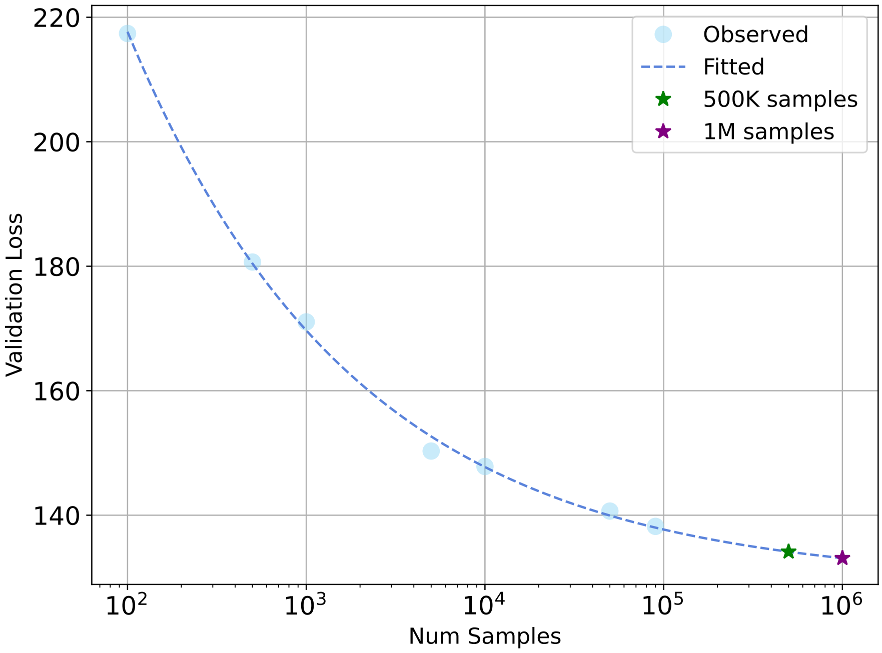
(c) Validation loss when fine-tuned on 10 samples over the number of samples in pre-training.

(d) Validation loss when fine-tuned on 100 samples over the number of samples in pre-training.

Figure 4.8: Investigating the downstream performance when pre-training DINOv2 on different datasizes.



(a) Curve fitted to the achieved validation losses, omitting the last value with 89972 samples. Stars denoted predictions made for 500k, and 1M samples. The red dot is the observed value for 89972 pre-training samples.



(b) Curve fitted to the achieved validation losses, including all observed values to get optimal predictions. Predictions made for 500k and 1M samples.

Figure 4.7: A curve has been fitted to the observed values to predict performance on larger datasets.

5

Conclusion

This section analyses and interprets the results of the pre-training methods, the scaling study, and the proposed bleeding fix approach. It also outlines potential directions for future work based on the observed findings and provides a summary of the thesis.

5.1 Discussion

This thesis has investigated the usage of vision foundation models as a base for pre-training a 3D object detector in LiDAR point clouds with promising results.

Notably, the pre-training methods does not affect the downstream performance at larger data sizes. As previously mentioned, both pre-training methods are web-scaled foundation models, meaning that they have been trained on vast and diverse image datasets sourced from the internet. This becomes especially impactful when fine-tuning on few samples, as the rich knowledge gained from pre-training is more valuable than the object information in a limited number of samples. However, as the size of the training dataset increases, the relative advantage of the web-scaled pre-training diminishes, since the model can learn specific features from the task-specific data.

We observe that until 10000 annotations are available for fine-tuning, both SAM 2 and DINOv2 pre-trained models outperforms the fully supervised model, proving its excellence on small number of annotated samples in fine-tuning. While the DINOv2 pre-training seems to provide a better ground for few-shot detection, SAM 2 outperforms DINOv2 for larger datasets when the backbone is kept frozen. We hypothesize that this is due to the SAM 2 pre-trained backbone outputting features more closely aligned with object detection but that it needs more data to extract that information. One reason for this could be that the transformer in the SAM-head does the actual work of resolving separate object and therefore when this part is removed during fine-tuning the new head require more data to learn this mapping. We further hypothesize that the DINOv2 pre-training provides a backbone feature map output which encodes the information in a more accessible way which is more suitable when fine-tuning on limited data. This could be an effect of the more suitable DINO-head which has been tailored to the DINO pre-training pipeline. Designing a more specific head for SAM could drive better performance for SAM fine-tuning. For example, using a transformer-based head could potentially help accessing the knowledge from pre-training in the backbone, possibly improving overall results. Lastly, we also note that pre-training DINOv2 is both more time and memory efficient in comparison to SAM 2, making it the more sustainable alternative.

Additionally, the transition from pre-training to fine-tuning is likely affected by the choices of learning rate and which layers are frozen during which stages of the training. A more systematic approach, as a parameter search, could help identify better configurations and enhance performance, particularly in balancing the transfer of general knowledge in the foundation models with adaptation to ZOD. The current results may have been influenced by suboptimal choices in these areas.

In relation to this, there are several other aspects of this project that could benefit form more carefully chosen parameters and design choices. These include the augmentations, NMS and IoU thresholds and voxel sizes. First of all the rotation angle for augmenting point clouds was selected based on our intuition rather than through optimization. Further experimentation with different values could offer better insight into their effects on the model's ability to learn and generalize. Secondly, the NMS and IoU thresholds both influence the interpretation of the predicted boxes. As this thesis primarily aims to compare the pre-training strategies, this does not affect the results, as the strategies remain comparable, but it is important to note that these threshold settings can bias the evaluation either in favour of or against the model, thereby influencing the perceived quality of its predictions. Lastly, the voxel size used in the model backbone to generate the pseudo image on which the ResNet runs can have a large effect on the performance of the model as a smaller voxel size would have provided the model with a more detailed picture at the cost of computational efficiency.

It is also worth mentioning that the DINOv2 and SAM 2 weights used in the implementation of this project was not taken from the largest available model which implicates that results could be improved by using larger foundation models.

Moreover, the mAP metric used to evaluate object detection performance in this project is based on calculating the IoU using AABB in the BEV space when obtaining the number of TP and FP. This represents a relatively generous measure of overlap, as the AABB of an object can cover a significantly larger area in BEV space compared to the oriented 3D bounding boxes predicted by the model and used in the ground truth annotations. While this may inflate the absolute mAP scores, we use high IoU thresholds which maintains a meaningful overlap requirement for true positives. Additionally, as all models are evaluated under the same conditions, the comparisons remain fair and meaningful.

The scaling analysis demonstrates that the DINOv2 pre-training indicate strong scalability. The pre-training performance, i.e. the validation loss during pre-training, follows a consistent and predictable trend as the number of training samples increases. This trend allows for reliable extrapolation to dataset sizes beyond those of ZOD. In contrast, the results from fine-tuning are less conclusive. They are characterized by considerable noise and a lack of clear trends in both mAP and validation loss, making it difficult to draw firm conclusions about scalability in the fine-tuning phase. The presence of numerous outliers suggests that fine-tuning dynamics, such as early stopping and parameter freezing, have a significant impact on the outcomes. Therefore, we leave that for future work to investigate further.

As the results show, the proposed bleeding fix significantly lowers the pre-training validation loss, indicating that removing points which as been assigned the wrong features helps the model learn more effectively. Furthermore, visualizations of the predicted feature clouds indicate that the model produces more reasonable predictions in regions where bleeding points have been removed. However, in downstream tasks, the mAP is notably lower when using the bleeding fix. We hypothesize that this is due to the removal of points with object features, which makes certain objects appear less frequently in the training data. In other words, the model has fewer opportunities to learn feature representations of concepts such as "vehicle." This outcome is somewhat unexpected and suggests that having a larger quantity of examples, even if some are noisy, may benefit performance more than focusing strictly on quality.

5.1.1 Future Work

The results demonstrate that both pre-training methods are highly effective in boosting model performance on one-shot and few-shot object detection. Therefore, future research may extend these results in several ways. In this section we describe key areas for further investigation that remained outside the scope of this project.

One direction of future work would be to introduce alternative ways to investigate the effectiveness of the pre-training strategies, beyond object detection. For example evaluating their impact on tasks such as semantic segmentation or , which could offer deeper insight into their generalization across diverse vision challenges.

In [13], the authors state that the lifted SAM masks suffers from edge-bleeding due to calibration errors. They further present a solution using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to cluster the 3D point cloud. If the overlap between a cluster and a 3D mask generated by SAM exceeds a certain threshold, the mask is replaced with the cluster. This approach reflects the intuition that a sufficiently accurate cluster is likely more reliable than a segmentation mask affected by calibration errors. Although this method was not implemented in this thesis, results from [13] suggest that it can enhance the quality of the masks used for SAM pre-training. This, in turn, may improve downstream performance in object detection tasks when utilizing SAM as a pre-training method.

The bleeding point removal algorithm proposed in this thesis was effective for boosting the pre-training performance with DINOv2, solving the issue of feature leakage from object points to background points. Future work includes both verifying that the bleeding fix solution performs well in downstream tasks and, in that case, this method could be explored as a means to reduce the edge-bleeding artifacts described in [13] and could be compared to that of the DBSCAN approach. While this poses an interesting area for further investigation, we hypothesize that its impact may be less pronounced when applied to the SAM 2 pre-training than for the DINOv2. This is due to the higher resolution of the 2D SAM 2 masks, which results in fewer 3D points being projected onto the same pixel and assigned the same value.

In addition, the [13] paper uses ground removal before generating the DBSCAN clusters. Since the DBSCAN clustering was not implemented in this thesis, the ground filtering was omitted. However, considering that the ground is typically considered background in object detection tasks, introducing ground removal before lifting the SAM masks and investigating its effect on downstream performance could be interesting for future work.

Another potential direction for future work, also inspired by [13], is to adopt a "FrankenFrustum" approach by extending the pseudo labels frustum range from 120° to 360°. Pre-training the model on a 360° field of view allows the model to be applied in the full point cloud as it ranges 360°. Another potential improvement is to apply CLIP tokens as semantic labels for the 3D masks, as demonstrated in [13]. This would allow for the generation of semantic segmentation pseudo-labels, in contrast to the binary masks in this project.

Lastly, an interesting avenue for future work is to separate the comparison between using feature clouds and segmentation masks as pseudo-labels from the choice of foundation model used in pre-training. This can be achieved by lifting intermediate features from SAM 2 into 3D space and using them for pre-training. Similarly, since DINOv2 has demonstrated competitive performance on semantic segmentation tasks [11], its segmentation masks can also be lifted to 3D and employed as pseudo-labels. Comparing this setup with the two pre-training methods discussed in this project would allow for a more controlled evaluation of the impact of label type versus the underlying foundation model.

5.2 Conclusion

This thesis presents how knowledge from the 2D domain can effectively be transferred to 3D object detection by utilizing web-scaled vision foundation models as pre-training methods. The results obtained show that pre-training with either SAM 2 or DINOv2 improves 3D object detection in terms of mAP compared to a fully-supervised baseline trained from scratch on limited labels. For one-shot detection specifically, pre-training with either SAM 2 or DINOv2 increases the mAP significantly, where DINOv2 achieves the best mAP of all three alternatives. These findings suggests that both pre-training methods are appropriate tools for reducing the reliance on annotated 3D data as they enable the model learn the properties of less common objects from only a few instances in which those objects are present.

In addition, this thesis proposes a solution to the bleeding present in DINOv2 feature assignment by bounding features to their correct object. The implemented solution demonstrably improves pre-training performance as less information is lost or corrupted.

The scaling study indicates promising scalability for DINOv2. However, further investigation is necessary, as the fine-tuning dynamics applied in this project might hinder a fair and sufficiently noise-free comparison.

To conclude, this thesis demonstrates the effectiveness of distilling knowledge from large vision foundation models into an object detector for improving few-shot detection performance and shows that it is feasible to use the output from vision foundation models as a pre-training signal for 3D object detection in LiDAR point clouds.

Bibliography

[1] Transportstyrelsen, *Statistik över vägtrafikolyckor*, Accessed: 2025-01-27, 2024. [Online]. Available: `https://www.transportstyrelsen.se/sv/om-oss/statistik-och-analys/statistik-inom-vagtrafik/olycksstatistik/statistik-over-vagtrafikolyckor/`.

[2] N. P. Gregersen and S. Forward, *Trafiksäkerhet: samspelet mellan människor, fordon och trafikmiljö*, 2nd ed. Stockholm: Norstedts Juridik AB, 2024, p. 403, ISBN: 9789139027652. [Online]. Available: `urn:nbn:se:vti:diva-20343`.

[3] S. Singh, *Critical reasons for crashes investigated in the national motor vehicle crash causation survey*, Feb. 2015. [Online]. Available: `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115`.

[4] Y. Chae, H. Kim, and K.-J. Yoon, "Towards robust 3d object detection with lidar and 4d radar fusion in various weather conditions," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 15 162–15 172. DOI: `10.1109/CVPR52733.2024.01436`.

[5] B. Wilson, Z. Kira, and J. Hays, *3d for free: Crossmodal transfer learning using hd maps*, 2020. arXiv: `2008.10592 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2008.10592`.

[6] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, *Point-bert: Pre-training 3d point cloud transformers with masked point modeling*, 2022. arXiv: `2111.14819 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2111.14819`.

[7] W. Shi and R. R. Rajkumar, "Self-supervised pretraining for point cloud object detection in autonomous driving," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 4341–4348. DOI: `10.1109/ITSC55140.2022.9922494`.

[8] R. Zhang, L. Wang, Y. Qiao, P. Gao, and H. Li, *Learning 3d representations from 2d pre-trained models via image-to-point masked autoencoders*, 2022. arXiv: `2212.06785 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2212.06785`.

[9] A. Radford, J. W. Kim, C. Hallacy, *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: `2103.00020 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2103.00020`.

[10] A. Kirillov, E. Mintun, N. Ravi, *et al.*, *Segment anything*, 2023. arXiv: `2304.02643 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2304.02643`.

[11] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, *Dinov2: Learning robust visual features without supervision*, 2024. DOI: `https://doi.org/10.48550/arXiv.2304.07193`. [Online]. Available: `https://arxiv.org/abs/2304.07193`.

[12] J. P. Huix, A. R. Ganeshan, J. F. Haslum, M. Söderberg, C. Matsoukas, and K. Smith, "Are natural domain foundation models useful for medical image classification?" In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 7619–7628. DOI: `10.1109/WACV57701.2024.00746`.

[13] A. Osep, T. Meinhardt, F. Ferroni, N. Peri, D. Ramanan, and L. Leal-TaixÃĺ, "Better call sal: Towards learning to segment anything in lidar," in *European Conference on Computer Vision (ECCV)*, 2024.

[14] M. Y. Lee, C. D. W. Lee, J. Li, and M. H. Ang, "Dino-mot: 3d multi-object tracking with visual foundation model for pedestrian re-identification using visual memory mechanism," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1202–1208, 2025. DOI: `10.1109/LRA.2024.3500882`.

[15] M. Caron, H. Touvron, I. Misra, *et al.*, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[16] N. Ravi, V. Gabeur, Y.-T. Hu, *et al.*, *Sam 2: Segment anything in images and videos*, 2024. arXiv: `2408.00714 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2408.00714`.

[17] S. Mallick. "Selective search for object detection." Accessed: 2025-05-13, LearnOpenCV. (2020), [Online]. Available: `https://learnopencv.com/selective-search-for-object-detection-cpp-python/`.

[18] J. Huppelen, *Selective search: Draft*, Accessed: 2025-05-13, 2025. [Online]. Available: `http://www.huppelen.nl/publications/selectiveSearchDraft.pdf`.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2014. arXiv: `1311.2524 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1311.2524`.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: `1506.02640 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1506.02640`.

[21] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016. arXiv: `1612.08242 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1612.08242`.

[22] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: `1804.02767 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1804.02767`.

[23] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection. arxiv 2019," *arXiv preprint arXiv:1904.01355*, 2019.

[24] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, Springer, 2020, pp. 213–229.

[25] X. Zhou, D. Wang, and P. Krähenbühl, *Objects as points*, 2019. arXiv: `1904.07850 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1904.07850`.

[26] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, Springer, 2016, pp. 483–499.

[27] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: `1512.03385 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[28] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, *Deep layer aggregation*, 2019. arXiv: `1707.06484 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/1707.06484`.

[29] G. Han and S.-N. Lim, “Few-shot object detection with foundation models,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 28 608–28 618. DOI: `10.1109/CVPR52733.2024.02703`.

[30] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[31] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[32] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[33] T. Yin, X. Zhou, and P. Krähenbühl, *Center-based 3d object detection and tracking*, 2021. arXiv: `2006.11275 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2006.11275`.

[34] J. Yan, Y. Liu, J. Sun, *et al.*, “Cross modal transformer: Towards fast and robust 3d object detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 268–18 278.

[35] X. Bai, Z. Hu, X. Zhu, *et al.*, “Transfusion: Robust lidar-camera fusion for 3d object detection with transformers,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1090–1099.

[36] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[37] Z. Liu, H. Tang, Y. Lin, and S. Han, “Point-voxel cnn for efficient 3d deep learning,” *Advances in neural information processing systems*, vol. 32, 2019.

[38] I. Misra, R. Girdhar, and A. Joulin, *An end-to-end transformer model for 3d object detection*, 2021. arXiv: `2109.08141 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2109.08141`.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: `1810.04805 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1810.04805`.

[40] E. Pot, A. Toshev, and J. Kosecka, “Self-supervisory signals for object discovery and detection,” *arXiv preprint arXiv:1806.03370*, 2018.

[41] A. W. Harley, Y. Zuo, J. Wen, *et al.*, “Track, check, repeat: An em approach to unsupervised tracking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 581–16 591.

[42] A. Osep, P. Voigtlaender, J. Luiten, S. Breuers, and B. Leibe, “Towards large-scale video video object mining,” *arXiv preprint arXiv:1809.07316*, 2018.

[43] M. Najibi, J. Ji, Y. Zhou, *et al.*, “Unsupervised 3d perception with 2d vision-language distillation for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8602–8612.

[44] E-spin, *Ethical considerations for data labeling and annotation (dla)*, Accessed: 2025-01-28, 2023. [Online]. Available: `https://www.e-spincorp.com/ethical-considerations-for-data-labeling-and-annotation-dla/`.

[45] D. Patterson, J. Gonzalez, Q. Le, *et al.*, *Carbon emissions and large neural network training*, 2021. arXiv: `2104.10350 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2104.10350`.

[46] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in nlp*, 2019. arXiv: `1906.02243 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1906.02243`.

[47] S. Sudhakar, V. Sze, and S. Karaman, “Data centers on wheels: Emissions from computing onboard autonomous vehicles,” *IEEE Micro*, vol. 43, no. 1, pp. 29–39, 2023. DOI: `10.1109/MM.2022.3219803`.

[48] T. Ercan, N. C. Onat, N. Keya, O. Tatari, N. Eluru, and M. Kucukvar, “Autonomous electric vehicles can reduce carbon emissions and air pollution in cities,” *Transportation Research Part D: Transport and Environment*, vol. 112, p. 103 472, 2022.

[49] A. P. Pandelu, *Day 47: Loss functions cross-entropy, mse, and mae*, Accessed: 2025-05-06, 2024. [Online]. Available: `https://medium.com/@bhatadithya54764118/day-47-loss-functions-cross-entropy-mse-and-mae-bdacebc7b428`.

[50] S. Fengde, “Understanding l1 and smoothl1loss,” *Medium*, 2023, Accessed: 2025-05-06. [Online]. Available: `https://someshfengde.medium.com/understanding-l1-and-smoothl1loss-f5af0f801c71`.

[51] H. Amit. “Implementation of dice loss vision pytorch.” Accessed: 2025-05-07. (2024), [Online]. Available: `https://medium.com/data-scientists-diary/implementation-of-dice-loss-vision-pytorch-7eef1e438f68`.

[52] K. Moore, N. Landman, and J. Khim. “Hungarian maximum matching algorithm.” Accessed: 2025-05-07. (2025), [Online]. Available: `https://brilliant.org/wiki/hungarian-matching/`.

[53] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R* (Springer Texts in Statistics), 1st ed. New York: Springer, 2013, ch. 6.3, p. 253, Chapter 6.3: Dimension Reduction Methods.

[54] M. Alibeigi, W. Ljungbergh, A. Tonderski, *et al.*, “Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[55] J. Kannala and S. Brandt, “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006. DOI: `10.1109/TPAMI.2006.153`.

[56] Q. Zhu, L. Fan, and N. Weng, “Advancements in point cloud data augmentation for deep learning: A survey,” *Pattern Recognition*, vol. 153, p. 110 532, Sep. 2024, ISSN: 0031-3203. DOI: `10.1016/j.patcog.2024.110532`. [Online]. Available: `http://dx.doi.org/10.1016/j.patcog.2024.110532`.

[57] J. Yang, K. Z. Luo, J. Li, *et al.*, *Denoising vision transformers*, 2024. arXiv: `2401.02957 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2401.02957`.

[58] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss, "Mask-based panoptic lidar segmentation for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1141–1148, 2023. DOI: `10.1109/LRA.2023.3236568`.

[59] M. A. Mazurowski, H. Dong, H. Gu, J. Yang, N. Konz, and Y. Zhang, "Segment anything model for medical image analysis: An experimental study," *Medical Image Analysis*, vol. 89, p. 102 918, Oct. 2023, ISSN: 1361-8415. DOI: `10.1016/j.media.2023.102918`. [Online]. Available: `http://dx.doi.org/10.1016/j.media.2023.102918`.

[60] M. Javaid, A. Haleem, R. P. Singh, and M. Ahmed, "Computer vision to enhance healthcare domain: An overview of features, implementation, and opportunities," *Intelligent Pharmacy*, vol. 2, no. 6, pp. 792–803, 2024, ISSN: 2949-866X. DOI: `https://doi.org/10.1016/j.ipha.2024.05.007`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2949866X24000662`.

[61] Z. Yan, W. Sun, R. Zhou, *et al.*, *Biomedical sam 2: Segment anything in biomedical images and videos*, 2024. arXiv: `2408.03286 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2408.03286`.

[62] B. Cui, M. Islam, L. Bai, and H. Ren, *Surgical-dino: Adapter learning of foundation models for depth estimation in endoscopic surgery*, 2024. arXiv: `2401.06013 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2401.06013`.

[63] H. Habehh and S. Gohel, "Machine learning in healthcare," 2021. DOI: `10.2174/1389202922666210705124359`. [Online]. Available: `https://pmc.ncbi.nlm.nih.gov/articles/PMC8822225/`.

A

Object Detection Examples

To illustrate the object detection performance of the different models Figures A.2, A.3, and A.4 illustrate the predicted and GT bounding boxes in BEV on a sample from the validation set after fine-tuning on different amounts of training samples. All predictions are generated using IoU threshold 0.1 within the NMS and objectness threshold 0.3. The legend in Figure A.1 relates to all object detection performance plots presented below.

Figure A.1: Legend related to object detection performance plots.

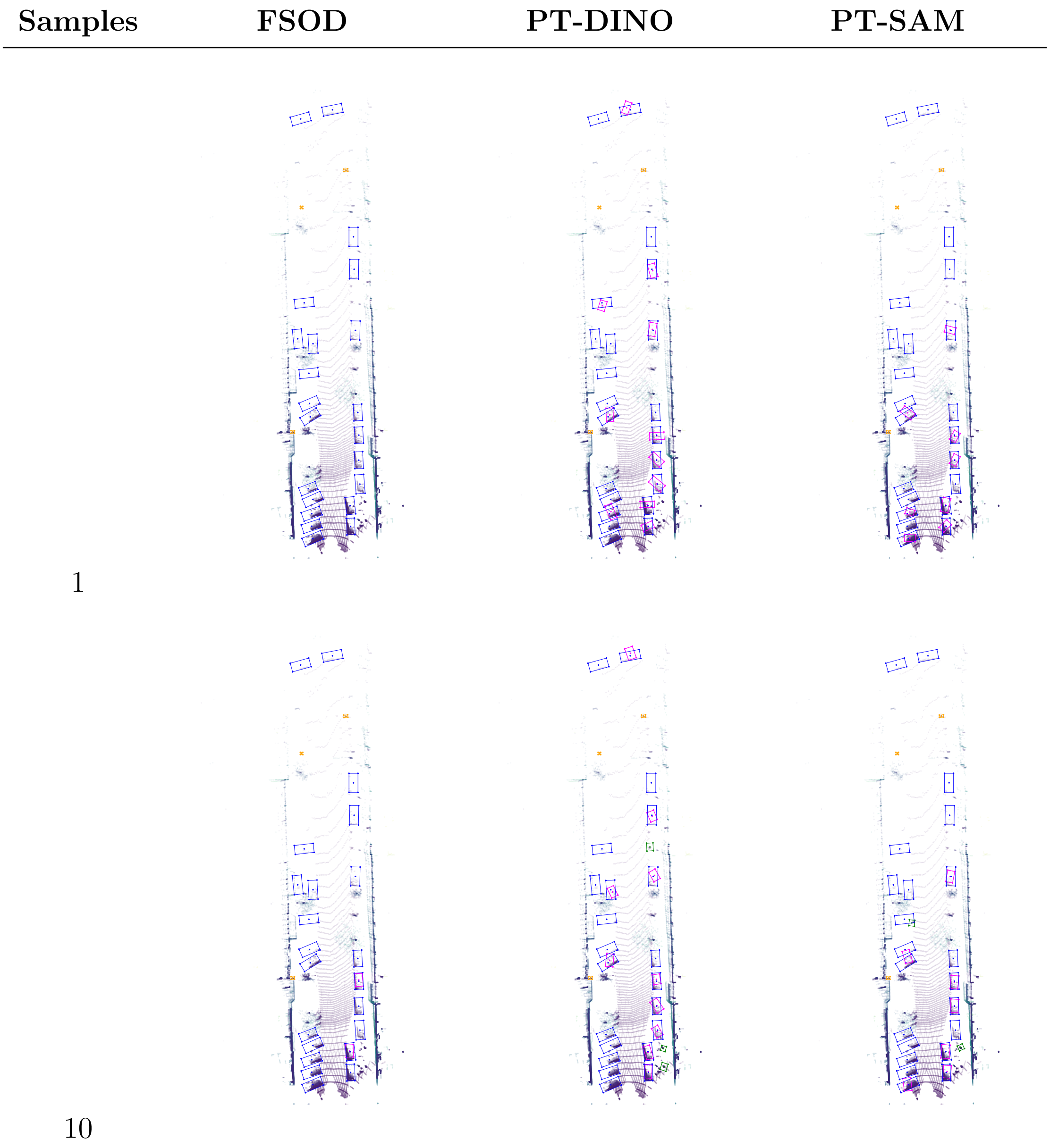


Figure A.2: Predicted versus GT bounding boxes fine-tuning on one and ten fine-tuning samples for the different models.

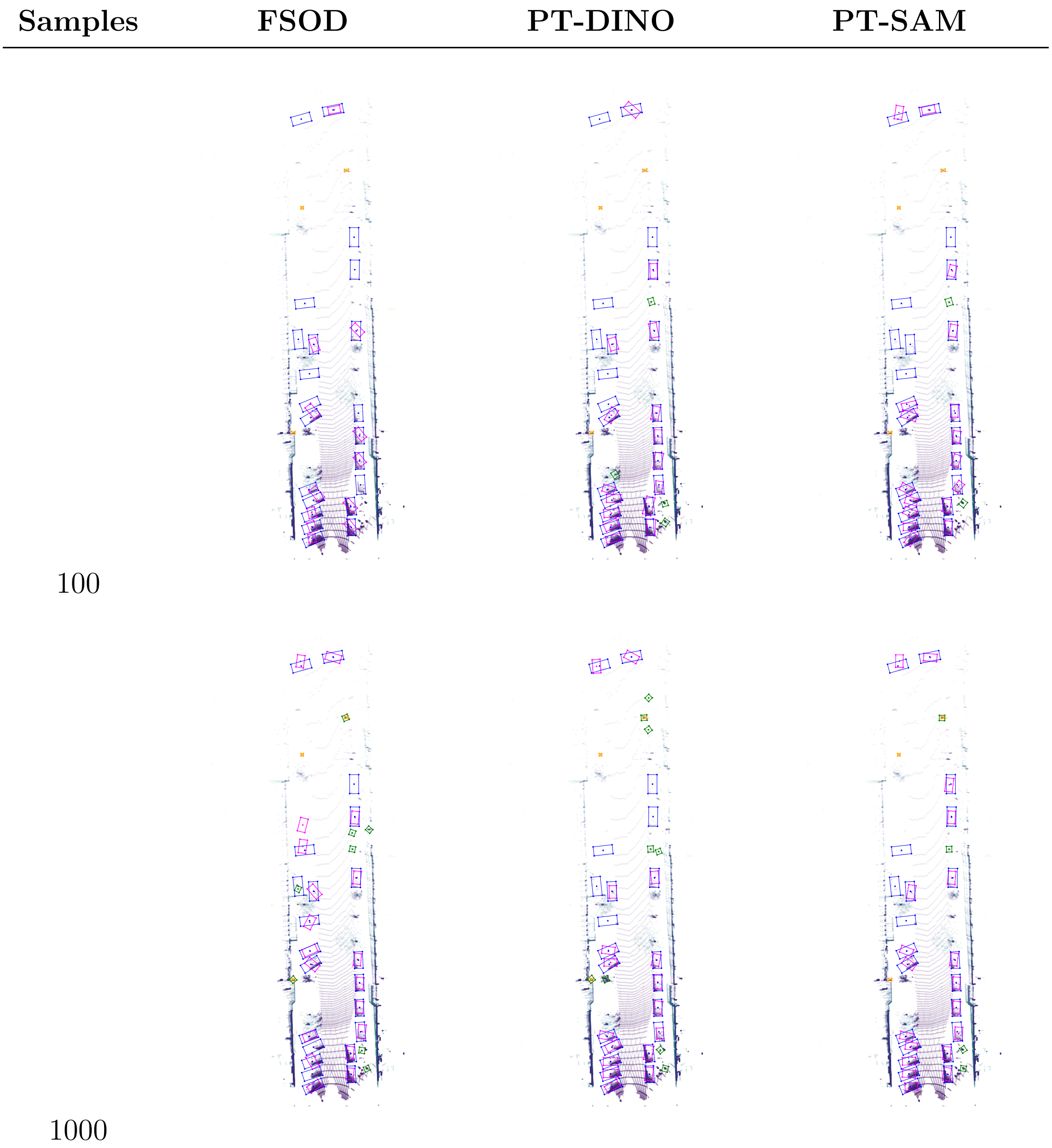


Figure A.3: Predicted versus GT bounding boxes fine-tuning on 100 and 1000 fine-tuning samples for the different models.

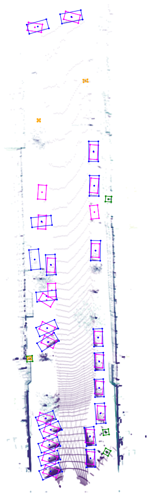
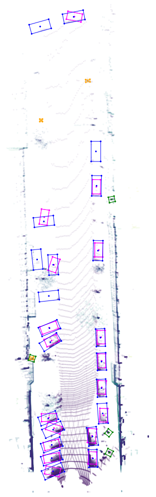
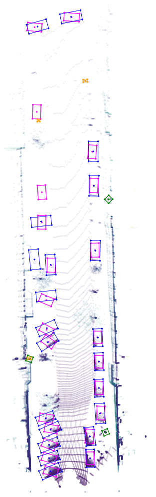
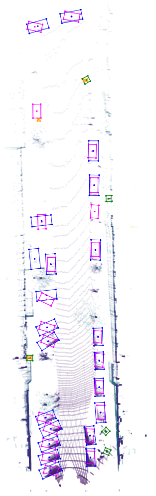
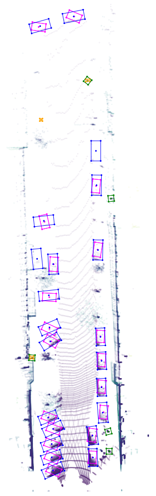
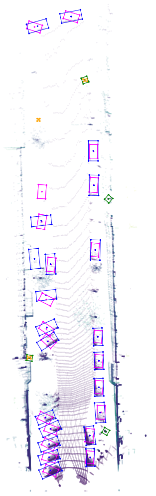
Samples	**FSOD**	**PT-DINO**	**PT-SAM**
10000			
89972			

Figure A.4: Predicted versus GT bounding boxes fine-tuning on 10000 and 89972 fine-tuning samples for the different models.

B

Application in Medical Domain

Due to one of the authors being enrolled in the Biomedical Engineering programme, this thesis includes a literature review of technologies similar to those developed in this project applied in the medical domain.

B.1 Introduction and Theory

This chapter provides an understanding of topics related to the project in a healthcare setting.

Incorporating Machine Learning (ML) in the medical domain faces several challenges, one of them being the lack of annotated data [59]–[61]. Another challenge is the domain change, as images and environments in the medical domain often have different characteristics compared to, for example, the automotive setting [59], [62], which this thesis focuses on.

B.1.1 Machine Learning in Health Care

ML is consistently transforming healthcare by enabling data-driven insights across diagnostics, treatment and assisting in decision making [63]. ML models can identify patterns and make predictions that support clinical decision-making, improving accuracy and decreasing the burden on medical personnel. Beyond that, ML applications have shown promising results in robotic healthcare applications, such as surgical assistance systems.

Despite these advancements, challenges persist. Ensuring data privacy and integrating ML tools into existing clinical workflows requires careful consideration [60], [63]. Moreover, the interpretability of ML models remains a critical factor for trust and adoption.

B.1.2 Computer Vision in Health Care

Computer Vision (CV) is also increasingly being integrated into healthcare to enhance diagnostic accuracy, treatment planning, and operational efficiency [60]. By enabling machines to interpret and analyse visual data, CV supports various medical applications including medical imaging, surgical assistance and patient monitoring.

In medical imaging, CV algorithms assist in detecting abnormalities in X-rays, MRIs and CT scans, allowing early diagnosis and reducing human error. For instance, CV systems can identify cancerous tissues with high precision, and has shown its excellence in identifying early symptoms of illness.

Lighting variation significantly impacts CV by altering how objects appear in images, making it different from human visual perception. Ongoing research and development aim to overcome hurdles such as privacy concerns and data quality, paving the way for more widespread adoption of CV technologies in medical practice.

B.1.3 Web-Scaled Foundation Models in Health Care

Recent research explores how web-scaled foundation models, like SAM, can generalise to the medical domain. The paper "Segment Anything Model for Medical Image Analysis: an Experimental Study" [59] investigates SAM's zero-shot performance on a range of medical imaging tasks, including radiology, histopathology, and dermatology. The findings suggest that while SAM shows potential in handling diverse imaging modalities, its performance often falls short of domain-specific models, particularly in segmenting subtle or complex anatomical structures. To address these limitations, the authors propose and evaluate several strategies, including prompt engineering and the potential for fine-tuning SAM on curated medical datasets. Similarly, "Biomedical SAM-2: Segment Anything in Biomedical Images and Videos" [61] investigates the potential of SAM 2 in a medical-specific setting, achieving promising results when fine-tuning SAM 2 on different medical image datasets.

Another direction in adapting foundation models to healthcare is illustrated by "Surgical-DINO: Adapter Learning of Foundation Models for Depth Estimation in Endoscopic Surgery." [62]. This study leverages DINO and adapts it for depth estimation in surgical endoscopy. Instead of conventional fine-tuning, the proposed method introduces lightweight adapter modules to fine-tune the pre-trained DINO model for endoscopic scenes, addressing the domain shift between general vision data and the complex, texture-scarce environment of surgery. The results demonstrate that adapter-based fine-tuning enables competitive or superior performance compared to training from scratch, with significantly reduced computational cost and data requirements.

B.2 Implementation

This chapter brings up the key implementation differences in the case of using similar methods to this project, based on the theory presented above.

In our project, SAM masks are generated by prompting the entire image with a grid, meaning that all visible objects can get segmented. In [59], they highlight the importance of carefully engineering the prompts when using SAM for medical tasks, as most often only one object in the image is of interest. Another reason is that it is specifically common in medical scenarios that an object is split, and non-contingous, for example, due to cross-sectional imaging methods. They also discuss the possibilities of making a medical-specified version of SAM, possibly fine-tuned, to improve the performance on medical tasks. Fine-tuning would be the more data-efficient approach.

In [62], the authors adapt the DINO model to the medical domain by freezing the image encoder and training only the integrated low-ranked adaptation layers and depth decoder. This approach allows them to specialise the pre-trained DINO model for depth estimation in endoscopic surgery. As a result, they create a version of DINO that is tailored to their specific task while maintaining the benefits of the original foundation model. In contrast to our project, which focuses on extracting features and generating pseudo labels without modifying the backbone model, their method involves direct optimisation of task-specific parameters within the DINO framework, making it a more integrated form of adaptation similar to one of the proposed methods in [59].

When developing this thesis, general precautions around personal data such as blurring faces were necessary, but the importance of this is even higher in healthcare due to privacy concerns.

B.3 Discussion and Summary

The results from this project prove the ability to transfer knowledge from the 2D domain to the 3D domain by effectively using web-scaled foundation models for pre-training. The results also indicate that one and few-shot detections are possible using the presented pre-training strategies in an AD/ADAS setting. If these are transferable to the medical domain, they offer a possible solution to the discussed lack of annotated data. Further investigation on the specific topic of health care has to be made to draw conclusions of the exact benefits of pre-training on web-scaled foundation models with medical fine-tuning, as the content of the web in relation to self-driving car scenarios may differ from the relation to health care-related images, but previous research ([59], [61]) shows that fine-tuning foundation models on domain-specific datasets improves the performance compared to solely relying on web-scaled data.

Based on the findings of [59], the performance of SAM in the medical domain is highly dependent on the characteristics of the dataset and the task. While SAM demonstrates strong performance in cases involving well-defined, clearly bounded structures, it tends to struggle with more complex or ambiguous scenarios. Nonetheless, its performance can be substantially improved when detailed prompts or additional guidance are provided. [62] has successfully demonstrated that it is possible to bridge the domain gap by adapting DINOv2 to the surgical domain. Their approach enables the model to be effectively used for tasks such as 3D reconstruction and surgical navigation.

Due to lighting variations, objects may appear different in an image compared to human perception. Our project demonstrates that pre-training on 2D image data can significantly improve 3D object detection performance on LiDAR point clouds.

This insight could be valuable in medical applications to be less vulnerable to image lightning artefacts, by incorporating other sensors while still being able to pre-train on more easily accessible camera data.