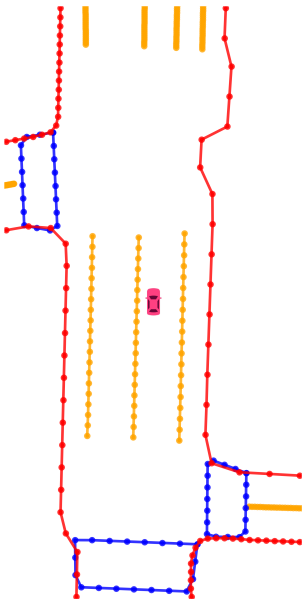


Semi-MapTR: Semi-Supervised Learning for End-to-End Online HD Map Construction

Master's thesis in Complex Adaptive Systems

Maximilian Salén & Axel Qvarnström

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Semi-MapTR: Semi-Supervised Learning for End-to-End Online HD Map Construction

Maximilian Salén & Axel Qvarnström

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Semi-MapTR: Semi-Supervised Learning for End-to-End Online HD Map Construction
Maximilian Salén & Axel Qvarnström

Supervisor: Lars Hammarstrand, Electrical Engineering
Advisor: Niklas Gustafsson, Jonathan Larsson, Adam Lilja, Rasoul Mojtahedzadeh. Zenseact
Examiner: Lars Hammarstrand, Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Top view of high-definition map.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Semi-MapTR: Semi-Supervised Learning for End-to-End Online HD Map Construction
Maximilian Salén & Axel Qvarnström
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Recently, the emergence of online high-definition (HD) mapping solutions for autonomous driving has shown great promise, such as MapTR [1] and VectorMapNet [2]. However, despite their success, these models depend heavily on large-scale annotated data for their training, thereby limiting their scalability due to the extensive manual labor required for data annotation.

This thesis addresses this critical challenge by investigating the integration of semi-supervised learning (SSL) into the online HD map construction framework. By leveraging a small portion of annotated data together with a larger portion of unlabeled data, the study aims to enhance the scalability and and efficiency of HD map generation. Therefore, we propose a semi-supervised approach built upon MapTRv2 [3], Semi-MapTR, utilizing pseudo ground truth generated by rasterizing vectorized output of unlabeled data and enhancing confidence through occupancy grid mapping.

We demonstrate an increase in both mean Average Precision (mAP) and mean Intersection over Union (mIoU) for both camera and LiDAR (Light Detection and Ranging) data using Semi-MapTR compared to MapTRv2, when both models are trained on the same amount of labeled data.

Keywords: deep-learning, semi-supervised learning, online mapping, HD maps, transformers, occupancy grid mapping, Bayesian statistics, inverse sensor model.

Acknowledgements

We would like to begin by expressing our sincere gratitude to our academic supervisor, Lars Hammarstrand. His patience, insightful advice, and support have been invaluable throughout this project. Even though we often left the early meetings feeling overwhelmed, we always walked away more knowledgeable, making it an exceptionally rewarding experience.

We would also like to extend our heartfelt gratitude to our supervisors at Zenseact: Jonathan Larsson, Adam Lilja, Niklas Gustafsson, and Rasoul Mojtahedzadeh. Their immense support and guidance have been instrumental in the success of this project. Without their invaluable assistance, the challenges we faced would have been far more difficult to overcome.

We want to give a special thanks to Jonathan Larsson for all the help he has provided throughout this project. His pedagogical approach and his readiness to assist during challenging times are truly are qualities you're looking for in a supervisor. We hope that he will continue to supervise and inspire other projects in the years to come.

Maximilian Salén & Axel Qvarnström, Gothenburg, June 2024

Contents

List of Figures

List of Tables

1
Introduction

Recent advancements in Autonomous Driving (AD) technology are captivating researchers and industry stakeholders due to promising improvements in transportation systems. These innovations in AD can enhance road safety and improve traffic flow efficiency, but also reduce costs associated with transportation, showcasing its critical role in shaping the future of mobility.

Despite the promising advancements, an inherent challenge within AD systems is the effective perception and interpretation of the surroundings. A very popular approach to this task is the application of Deep Neural Networks (DNNs) alongside data from sensors such as cameras and LiDAR (Light Detection And Ranging). However, DNNs frequently encounter overfitting due to the sheer number of trainable parameters, which accentuates the use of large datasets for their training. This, in turn comes with another issue, which is the manual labor required to annotate such datasets. Naturally, approaches that rely on less of this kind of data have gained a lot of attention. One such approach is Semi-Supervised Learning (SSL), a method that only requires a small amount of annotated data together with a large amount of unannotated data. This permits leveraging a large dataset, with just a fraction requiring annotation.

In later years there's been a shift towards high-definition (HD) map representations in AD, which provide crucial and accurate guidance for navigation and decision-making. The advent of online end-to-end vectorized HD map constructors in autonomous driving has introduced highly detailed and precise environmental representations, proving to be a groundbreaking advance, as demonstrated by recent studies [2, 1, 3, 4, 5, 6]. A main task for online HD mapping is to provide real-time creation and updates of HD maps using onboard sensor data, as opposed to traditional HD mapping, which creates HD maps in advance and cannot adapt to changes in the traffic environment. Another important task for online HD mapping is addressing the issue of scalability, as online HD mapping alleviates the vast amount of human effort involved in annotating HD maps. However, these models are commonly trained and evaluated on large datasets, such as nuScenes and Argoverse2 [7, 8], which continues to present significant challenges for scalability. This shortcoming reveals an interesting research opportunity closing the gap between SSL and online end-to-end HD map construction. Therefore, this research will explore the integration of SSL within these advanced HD map construction models, aiming to fill the void of such knowledge, hoping to enhance the scalability and precision of autonomous systems.

1.1 Purpose

With the advancement of end-to-end learning in HD map construction, as discussed in the previous section, and the success of recent works employing semi-supervised methods [9, 10, 11], integrating SSL into the HD map construction framework seems to be a natural progression. SSL not only has the potential to improve scalability and reduce the need for large scale annotated data, but also improve precision and generalization of HD map constructors, particularly in scenarios where labeled data is scarce but unlabeled data is abundant. Thus, the purpose of this thesis is to evaluate the integration of semi-supervised learning into online HD map construction. The research will be guided by the following questions:

- How does SSL improve the precision and generalization of HD map constructors compared to fully supervised methods?
- How do camera data and LiDAR data impact the precision and generalization capabilities of HD map constructors when using semi-supervised learning?

1.2 Thesis Outline

This thesis is organized into six main chapters, each addressing a specific aspect of the research. Here follows an outline of the structure and the content of each chapter:

Chapter 1: Introduction
Chapter 1 introduces the thesis, creating a research space as well as an overview of the research topic. It also states the purpose of thesis and explains the significance of the research together with the scope of the study.

Chapter 2: Background
In chapter 2 follows a background, necessary for understanding the key aspects of the research; online HD mapping, semi-supervised learning and occupancy grid mapping. It begins by introducing the fundamental concepts related to these topics. Following this, a review of the existing literature pertinent to both online HD mapping and semi-supervised learning is presented. Additionally, a detailed description of the foundational model used in this project, MapTRv2, is included.

Chapter 3: Methods
This chapter describes the methods used in this thesis, specifically how pseudo labels to be used in the SSL-process is created, the approach to achieving confidence in these labels using occupancy grid mapping, and the supervision techniques for training the network.

Chapter 4: Results
Chapter 4 presents the results from our experiments, comparing our contribution to the foundational model of this project, MapTRv2. Furthermore, we also present results of several ablation studies, showcasing the most important parts of the model

and how they contribute to the model's performance.

Chapter 5: Discussion
In Chapter 5, a discussion following the findings from chapter 4 is presented, specifically the results on adding semi-supervision to MapTRv2, an evaluation of the creation of pseudo ground truth and the findings of the ablation studies.

Chapter 6: Conclusion
In the final chapter, we conclude the thesis by summarizing the key findings from our research and suggesting areas for further investigation.

1.3 Delimitations

This project will primarily focus on the development and evaluation of applying SLL to an existing end-to-end deep learning model using the nuScenes dataset. Thus, no modification will be made to the model architecture and it will be used as it is. Furthermore, since a part of the data needs to be set aside for the unlabeled portion of the training, we are not going to benchmark against already existing results of SOTA models produced on the full nuScenes dataset.

1.4 Contributions

The main contribution of this work is to provide a first look at the integration of a semi-supervised learning approach into a state-of-the-art framework for online HD map construction, specifically the MapTRv2 framework [3]. The results shows that:

- Adding semi-supervision to MapTRv2, improves it's performance but most importantly improves the segmentation capabilities. Notably, the ability to predict pedestrian crossings has seen substantial improvement, especially when using camera data.
- Using an occupancy grid mapping to create pseudo ground truth is an effective method for leveraging unlabeled data in the context of online HD mapping.

2
Background

This chapter provides an overview of the theories behind the methods used in this thesis. It begins with an overview of early HD map construction models, highlighting their limitations. Subsequently, the chapter progresses towards end-to-end learning based construction of HD maps, addressing these short-comings. It introduces the foundational model used in this project, MapTRv2 and its predecessor MapTR, providing detailed insights into their architecture and functionality. After that, a brief introduction to semi-supervised learning is provided, followed by a discussion of popular semi-supervised learning methods and the current literature of the topic. Finally, the chapter concludes by presenting theory on occupancy grid mapping, providing context for its relevance and applications within the scope of this thesis.

2.1 Online HD Map Construction

The task of online HD map construction aims to dynamically construct the local semantic map using onboard sensor observations. A local HD map can be described by a set of map elements belonging to different categories, e.g. pedestrian crossings, lane dividers, road boundaries etc. Each map element can be vectorized to a polyline, consisting of a set of points and can thus deal with complicated and even irregular road structures. In Figure 2.1, an example of a topdown view of an HD map is illustrated.

One of the first models that construct these in a dynamic way is HDMapNet outlined in [12]. With the use of an image encoder and a LiDAR point cloud encoder onboard sensory observations from both camera and LiDAR is encoded into a bird's eye view (BEV) representation which is fed to a BEV decoder. This decoder then predicts a semantic segmentation, instance embedding and lane direction. Finally a post-processing step is applied to cluster instances from embeddings and vectorize them. However, due to HDMapNet relying on rasterized map predictions and a heuristic post-processing step the model's scalability and performance is restricted. In the following section novel approaches to overcome this restriction is discussed.

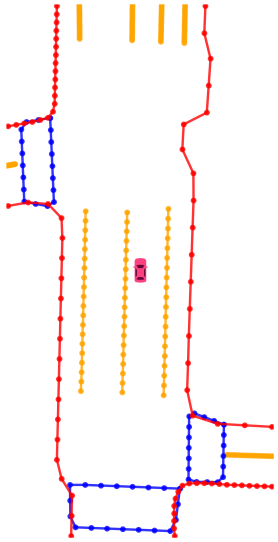


Figure 2.1: Top-view of an vectorized HD map. Road boundary (red), lane divider (yellow), pedestrian crossing (blue).

2.2 Online HD Map Construction with End-to-End Learning

To overcome the restrictions of post-processing steps, MapTR oulined in [1] presents an efficient way of constructing online vectorized HD maps using an structured end-to-end transformer. This novel approach is an advance for autonomous vehicles. It creates online structured HD maps which gives precise environmental information about the driving scene which is an important component for planning and navigation in autonomous vehicles. In contrast to HDMapNet it is a direct set prediction model which removes many hand-designed components for post processing methods such as clustering. This makes the construction of HD maps more generalized, meaning it is more adaptable varying situations.

MapTR adopts a DETR-like paradigm, where DETR [13] is a encoder-decoder transformer architecture for end-to-end object detection. Differently from traditional object detection where objects are geometrically abstracted as bounding boxes, MapTR models each map element as a point set, effectively reframing the online HD map construction task as object detection. However, [14] highlights how this approach fails to take global semantic information into account. To address this limitation the authors propose a segmentation-guided structured model for online HD map construction. This approach, named MapSeg, incorporates a BEV segmentation module among other enhancements based on the MapTR structure. This has enabled the model to better capture the semantic information, which highlights how MapTR can be improved by incorporating semantic information, something which is done in its later iteration MapTRv2.

2.2.1 MapTR Architecture

The architecture of MapTR consists of a map encoder, a BEV transformation, and a map decoder. The map encoder is responsible for extracting features from sensor inputs, which can include camera images, LiDAR point clouds, or a combination of the two. These extracted features are then transformed into a BEV representation. The map decoder employs hierarchical queries to explicitly encode each map element, working at both point level and instance level for detailed map construction. The prediction head of the system includes a classification branch, responsible for determining the class of each instance. Additionally, there's a regression branch designed to precisely calculate the positions of the points constituting each map element. An overview of the overall architecture of MapTR is depicted in Figure 2.2.

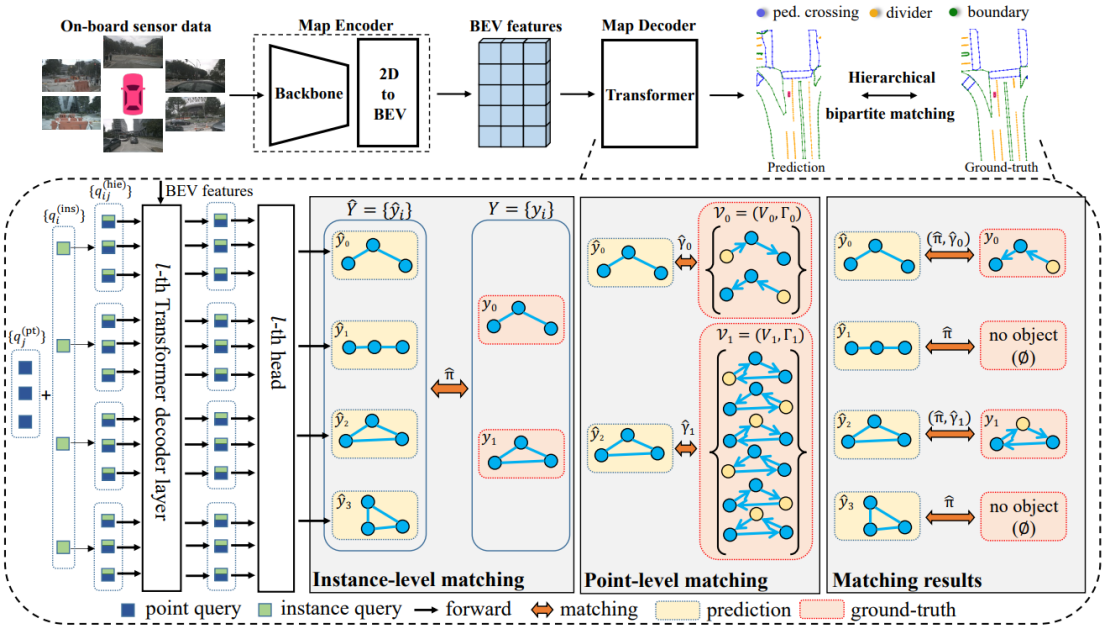


Figure 2.2: The overall architecture of MapTR. MapTR adopts an encoder-decoder paradigm. The map encoder transforms sensor input to a unified BEV representation. The map decoder adopts a hierarchical query embedding scheme to explicitly encode map elements and performs hierarchical matching based on the permutation-equivalent modeling. MapTR is fully end-to-end. The pipeline is highly structured, compact and efficient. Image and figure description reprinted from [1]. Licensed under CC BY-NC-SA 4.0.

2.2.2 Permutation-Equivalent Modeling

During the training of MapTR, points are sequentially sampled at the boundaries of the map elements' shapes. Elements with open shapes are transformed into polylines, while elements with closed shapes are transformed into polygons. MapTR uses these point sets in a hierarchical bipartite matching process. Specifically, it

performs instance-level matching followed by point-level matching. The purpose of instance-level matching is to assign the correct element label to the point set.

Once the instance-level matching is completed, the point-level matching assigns the correct shape to the corresponding ground truth element shape. Since the point set of each map element represents the vertices of its shape (boundary points), the order in which the points are connected matters, and this is where point-level matching comes in. For example, given a three-point polygon, there are six ways to order (permutations) the edges between the points, as illustrated in Figure 2.2 (Point-level matching box). Thus, different permutations can represent different shapes for the same set of points.

The point-level assignment seeks to find the permutation that best aligns with the ground truth shape. This permutation representation of a map element addresses the ambiguity of ordering. If the model were only trained to recognize one specific ordering of points, it might fail to recognize the same map element when represented in a different order. Therefore, the bipartite element matching in the decoder of the transformer ensures the model can handle various permutations effectively.

2.2.3 MapTRv2

We base our contribution in this project on the later iteration of MapTR, known as MapTRv2, introduced in [3], which includes several improvements over its predecessor. Some of which follows below:

- MapTRv2 performs decoupled self-attention, which separately executes attention operations along two dimensions: inter-instance and intra-instance, allowing for more focused and efficient processing of map data. This enhances the computation speed and is more memory efficient than using the vanilla self-attention used in MapTR. Then cross-attention is used to make map queries interact with the input features.
- MapTRv2 performs one-to-many set predictions instead of one-to-one, which speed up the convergence of the training process.
- Another improvement is the use of auxiliary dense supervision for bird's eye view and perspective view (PV). This means that the model during training receives detailed (dense) feedback information, and the use of both BEV and PV makes the model learn a more accurate and robust representation of the environment.

The key improvement here is MapTRv2 inclusion of auxiliary dense supervision, which in order to futher leverage semantic and geometric information introduces three auxiliary dense prediction losses

$$\mathcal{L}_{\text{dense}} = \alpha_d \mathcal{L}_{\text{depth}} + \alpha_b \mathcal{L}_{\text{BEVSeg}} + \alpha_p \mathcal{L}_{\text{PVSeg}}. \tag{2.1}$$

Our primary interest lies in the BEV segmentation loss rather than the depth prediction or PV segmentation losses. Rasterized ground truth (GT) maps on the BEV canvas generate a foreground mask, and the BEV segmentation loss is calculated as

the cross-entropy loss between the predicted BEV foreground-background segmentation map and the binary GT map mask. Later in this report, we will show how semi-supervision can be applied in a similar manner to the BEV segmentation loss.

2.3 Semi-Supervised Learning

Semi-supervised learning is very similar to traditional supervised learning in the way that the aim is still to approximate a function f that maps X to Y. The main difference lies in leveraging unlabeled data together with labeled data, which we denote U and L respectively. In almost every case unlabeled data is much easier obtained than labeled data, thus it is common that $|U| \gg |L|$. An overview of semi-supervised learning is shown in Figure 2.3.

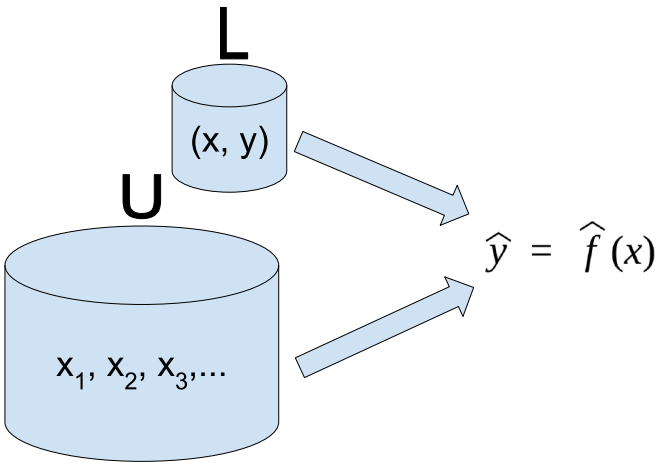


Figure 2.3: A schematic illustration of semi-supervised learning.

A challenging aspect of semi-supervised methods lies in constructing their loss function. While supervised learning benefits from ground truth labels, enabling the use of cross-entropy loss, the absence of labels in semi-supervised scenarios renders cross-entropy inapplicable, at least at first sight. Today there exists many variations of semi-supervised learning methods which effectively deals with this challenge, some of which will be discussed in the subsections that follows.

2.3.1 Pseudo-labeling

As foreshadowed earlier, there's a method to utilize cross-entropy loss in semi-supervised learning without excessive complexity. In this method, also known as pseudo-labeling, model predictions are used to create labels for the unlabeled data also referred to as pseudo-labels. These pseudo-labels and their corresponding samples are then added to L, being treated as ground truth. This of course comes with the risk of incorrect labels, a key challenge with self-supervision.

An early pseudo-labeling method first presented in 1998 by [15] proposed co-training, where the idea is to use two views of the same object to train two different learners.

Thereafter, both learners predict the labels of unlabeled data and adds the most confident predictions in to L. This approach enables a co-learning framework where both learners can learn from each other.

Another approach that significantly popularized the pseudo-labeling method was introduced in [16]. This method involves creating pseudo-labels by assigning the class with the highest predicted probability as the label for each unlabeled instance during each weights update. These pseudo-labels are then treated as true labels in subsequent training iterations.

More recently, [17] presented a naive semi-supervised deep learning approach using pseudo-labels. This method uses the output from a network on unlabeled data as pseudo labels. Thereafter, the pseudo labels are used to pre-train the deep learning model and fine-tune it using the labeled data. This combination of pseudo-labeling, pre-training and fine-tuning is called naive semi-supervised deep learning.

2.3.2 Consistency Regularization

Consistency Regularization is a technique which aims to improve model robustness by ensuring accurate predictions on labeled samples while being consistent on unlabeled samples. This is achieved by applying different augmentations to the input data such as noise, rotation or blurring and then training the model to predict the same output for both the original and the augmented versions. In this method there's no need for ground truth labels, as the main objective is to achieve prediction consistency between original and modified data.

This concept was first introduced by [18] and later popularized by [19, 20]. They proposed an unsupervised loss function, that penalizes the network when it produces inconsistent results for the same input under varying augmentations. Another method which draws from both consistency regularization and pseudo-labeling is FixMatch [10]. Using both of these techniques the algorithm creates artificial labels. The algorithm weakly augments an unlabeled sample (image) using e.g. flip-and-shift augmentation, and then uses the labels produced for this sample as a target for a strongly augmented version of the same sample. They propose several augmentation tools for achieving heavily-distorted versions of a given image. In the end, only labels that are assigned a high probability to one of the possible classes are retained. An example diagram of FixMatch is shown in Figure 2.4.

2.3.3 Student Teacher Framework

The student-teacher framework is an alternative approach that has recently shown success in the works of [20, 21, 9]. As the name suggests, this framework involves a teacher network guiding a student network by producing training targets for the student to learn from. However, teachers and how targets are applied when training the student can be implemented very differently. One way to do this is by Temporal Ensembling [20], in which average predictions from the last n students

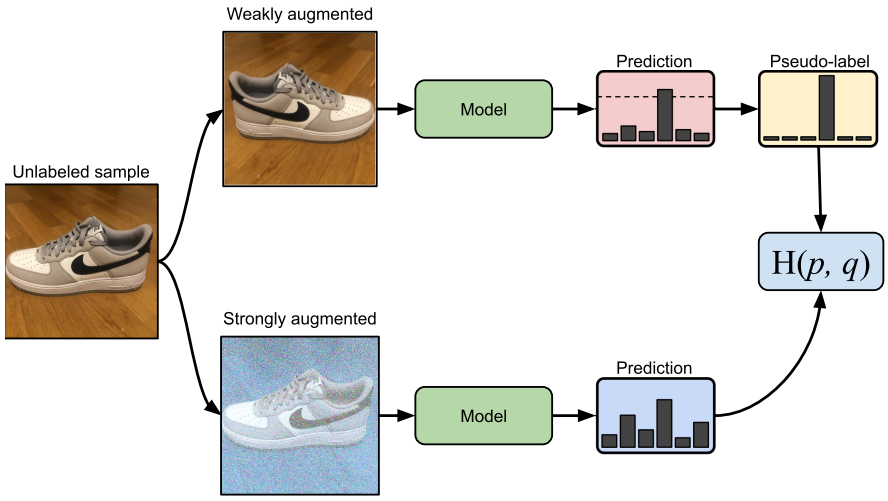


Figure 2.4: Example diagram of FixMatch. Given an unlabeled sample, the algorithm weakly augments the image and feeds it to the model to produce a prediction (red box). Once the model assigns a probability to any class which is above a certain threshold (dotted line), said prediction is converted to a one-hot pseudo label. The algorithm proceeds by strongly augmenting the same sample (bottom). The models learns to match its prediction on the strongly-augmented version with the pseudo-label using a cross-entropy loss function.

models are used to produce the targets. Alternatively, Mean Teacher [21] creates the teacher by calculating the mean networks weights over the last n student models.

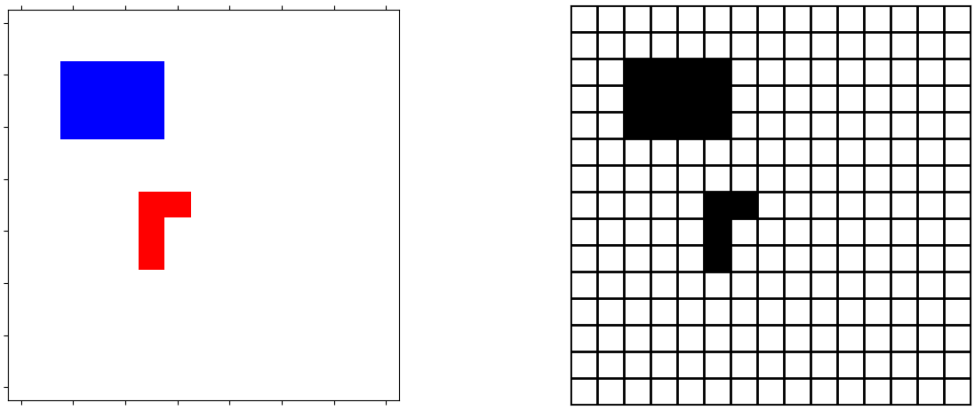
Recently, a novel method for semi-supervised object detection (SSOD) which combines object detection using the DETR-framework [13] and the Teacher-student architecture was presented in [9]. Following the principles of consistency regularization, weak and strong augmented unlabeled images are fed to the teacher and student respectively. Pseudo-labels created by the teacher which has a confidence score above a threshold are used to supervise the student. Then, using back-propagation the weights of the student model are updated while the teacher model's parameters are updated using a exponential moving average of the student.

The original DETR-framework [13] relies heavily on one-to-one assignment which imposes a high risk of generating sparse low-quality proposals when using it in conjunction with semi-supervised object detection (SSOD). This is where the authors propose to replace one-to-one with one-to-many assignment in the early stage of training in order to effectively exploit multiple positive queries to realize semi-supervised learning. By assigning multiple positive proposals to each pseudo label, high-quality positive proposals are given the opportunity for optimization. This approach significantly enhances the convergence speed and, as a result, yields pseudo

labels of superior quality. However, having multiple positive proposal for each pseudo label does come at a cost, it results in duplicate predictions. To mitigate this problem, the second stage reverts to one-to-one assignment training. This ensures the enjoyment of high-quality pseudo labels after the first stage of training, followed by a gradual reduction of duplicate predictions to eventually achieve a non-max suppression-free detector with one-to-one assignment training in the second stage. Additionally, the model incorporates both a consistency regularization approach and a cost-based pseudo label extraction technique, enhancing its ability to recognize stable, distinctive attributes across various augmented views.

2.4 Occupancy Grid Mapping

Occupancy grid mapping, introduced by Elfes [22], involves the use of a two-dimensional grid divided into cells, where each cell have a probability of being either occupied or empty. The occupancy grid has a spatial resolution which determines the area each cell covers. With a higher resolution, the grid will get more cells, each covering a smaller area meaning a more detailed information. The state of the occupancy grid will be updated with a sensor model which includes sensor information. An example of an occupancy grid is depicted in Figure 2.5.



(a) Real environment **(b)** Occupancy grid of environment

Figure 2.5: Simple example how an occupancy grid in (b) of the real environment in (a) looks.

2.4.1 Inverse Sensor Model

The inverse sensor model, $p(x|z_t)$, models the probability that a cell x is occupied given the sensor measurement z_t. In other words, it defines the probability of a state when given a sample at timestamp t. The measurements from a sample at t is all the predictions, which comes from all the sensor readings when the vehicle is at time step t. This inverse sensor model is then used in a Bayesian update to build the occupancy grid for a scene.

2.4.2 Bayesian Update

The Bayesian update is a crucial step in the occupancy grid mapping, refining each cells estimates when new measurements are available. [23] derive all the steps defining the update but we will only cover the essential ones:

We start with a bayes filter

$$p(x|z_{1:t}) = \frac{p(z_t|x, z_{1:t-1})p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})}, \tag{2.2}$$

where $z_{1:t-1}$ denotes all measurements from time step 1 to $t-1$.

Using the Markov chain assumption ($p(z_t|x, z_{1:t-1}) = p(z_t|x)$), we get

$$p(x|z_{1:t}) = \frac{p(z_t|x)p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \tag{2.3}$$

Then, we expand the equation using Bayes' rule, ending up with

$$p(x|z_{1:t}) = \frac{p(x|z_t)p(z_t)}{p(x)} \cdot \frac{p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \tag{2.4}$$

Thus, we have turned the forward sensor model $p(z_t|x)$ in (2.3) into an inverse sensor model $p(x|z_t)$.

Now, to derive the probability of a cell being empty $p(\bar{x}|z_{1:t})$, we follow the same steps as above and obtain

$$p(\bar{x}|z_{1:t}) = \frac{p(\bar{x}|z_t)p(z_t)}{p(\bar{x})} \cdot \frac{p(\bar{x}|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \tag{2.5}$$

Next, we divide the update rules (2.4) with (2.5) and get

$$\frac{p(x|z_{1:t})}{p(\bar{x}|z_{1:t})} = \frac{p(x|z_t)}{p(x)} \frac{p(\bar{x})}{p(\bar{x}|z_t)} \frac{p(x|z_{1:t-1})}{p(\bar{x}|z_{1:t-1})}. \tag{2.6}$$

Finally, we apply the logarithm on (2.6) to avoid numerical instability, hence the final update rule is the log odds of the belief that the cell is occupied over the belief that the cell is empty:

$$l_t(x) = \log \frac{p(x|z_t)}{p(\bar{x}|z_t)} + \log \frac{p(\bar{x})}{p(x)} + l_{t-1}(x). \tag{2.7}$$

Here the first term is the log odds of cell x being occupied or not, given z_t. The second term represents the prior knowledge about the cells occupancy. The third term represents the log odds from the previous state, which accumulates information over time. With this formula, we have a way to iteratively update the log probability of any given cell of the occupancy grid map with new measurements.

3
Methods

In this chapter, we are going to present the techniques employed in this project, starting with an overview of the dataset and of the proposed method. We will then provide a detailed explanation of how the pseudo ground truth is generated from MapTRv2 predictions. Moving forward, we will discuss the supervision component, explaining how semi-supervision can be achieved utilizing MapTRv2's auxiliary BEV segmentation head. Lastly, we will outline the training loop and provide details of the training process, experimental setup, and evaluation metrics.

3.1 Dataset

The dataset utilized to generate the results for this study is the nuScenes dataset [7], which is a public large-scale dataset for autonomous driving. It consists of in total 1000 driving scenes collected from Boston and Singapore, known for their dense traffic and difficult driving conditions. Each scene is roughly a 20 second long sequence of samples collected from multiple sensors. Key samples are annotated at 2Hz and contains 6 camera images, all of different views, and LiDAR sweeps. Our dataset pre-processing steps mirror those of MapTR [1] exactly, incorporating three categories of map elements from the nuScenes dataset: pedestrian crossing, divider, and road boundary.

3.1.1 Dataset Preproccesing

Following the findings in [24], which highlighted significant data leakage in the nuScenes dataset due to extensive geographical overlap between the training, validation, and test sets, we will adopt the dataset split proposed by the authors. Their approach involves dividing the nuScenes dataset based on sample positions, known as Geographically Disjoint splits. This method has demonstrated that SOTA models, including MapTRv2, perform considerably worse—some experiencing a drop of up to 45 mAP—when trained and evaluated on these proposed splits.

Additionally, to reduce the training time, the data will be subsampled to a frequency of 1 Hz instead of the original 2 Hz. This adjustment is considered reasonable as the model will still observe most of the surrounding environment even with the subsampled data.

A portion of the data must also be set aside for the unlabeled segment of the semi-

supervised training. We chose to achieve this by randomly sampling from the data, as it is the most straightforward approach. This method is particularly applicable in real-world scenarios where large amounts of unorganized, unlabeled data are mixed with some labeled data.

3.2 Overview of Proposed Method

We propose a novel method which encompasses the techniques employed in self-supervision, i.e. pseudo-labeling, and occupancy grid mapping within the MapTRv2 architecture. Using the original architecture of MapTRv2 we initially train the model supervised on a portion of the dataset. Thereafter, we let the model make predictions on the remaining unlabeled data, which are rasterized and used to create scene maps through occupancy grid mapping, using the confidence score for the different map elements provided by the classification branch of MapTRv2, thus constructing probabilistic occupancy grids which will be used as pseudo ground truth.

Now, having the pseudo GT created, we use it for supervising MapTRv2's auxiliary BEV segmentation head, but replacing the foreground-background segmentation with semantic segmentation for the different map classes. This loss is then added to the dense prediction loss, successfully incorporating semi-supervision into the MapTR framework.

3.3 Creating Pseudo Ground Truth

In this section, the method for creating the pseudo ground truth is presented. It begins with transforming each sample of a scene into a common coordinate frame and then proceeds to rasterize the data into a grid representation.

3.3.1 Coordinate Transformations

MapTRv2 generates 50 map elements per sample, each consisting of a polyline with 20 points, a class label, and a confidence score. Utilizing the LiDAR pose from the sample information provided by the nuScenes dataset, each scene can be reconstructed into an HD map. This is achieved by transforming the map elements of the samples that belong to a scene into a common coordinate frame. Although the common frame is typically chosen to be the first sample in each scene, any sample could serve this purpose.

The point sets making up the map elements in each sample are defined in relation to the LiDAR coordinate system within a range of $x = [-15, 15]$m and $y = [-30, 30]$m. The transformation needed to relate the points (T) to the common coordinate frame is defined by (3.1), requiring a relational transformation matrix consisting of rotation (R) and translation (t).

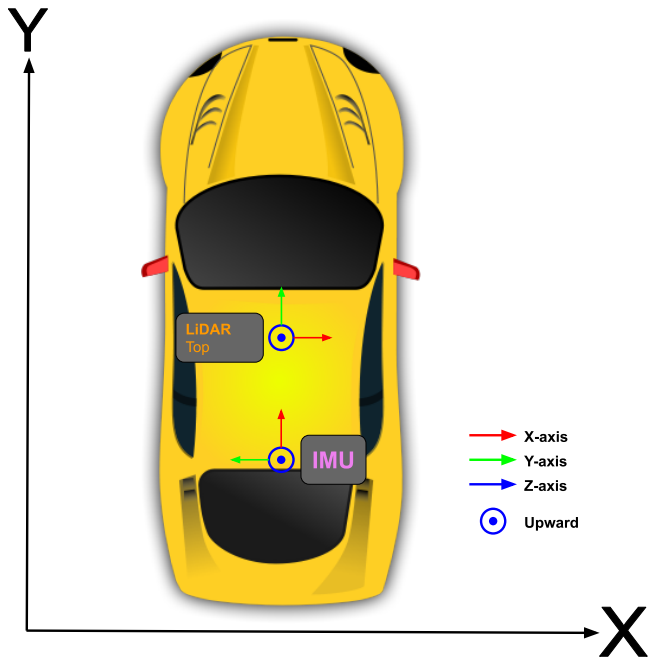


Figure 3.1: LiDAR and IMU position and orientation for ego vehicle in nuScenes dataset. Adapted work of Peeyush.ciit, CC BY-SA 4.0, via Wikimedia Commons.

$$\begin{bmatrix} y' \\ x' \\ z' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.1}$$

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

The nuScenes dataset provides two specific coordinate relations, how the LiDAR is positioned in relations to the ego vehicle, i.e. the inertial measurement unit (IMU), and how the IMU is related to a global coordinate system, see Figure 3.1. Given these two relation, a third can be constructed as the relation between the LiDAR position in the global coordinate frame as in (3.3).

$$T_{\text{Global, LiDAR}} = T_{\text{Global, Ego}} \cdot T_{\text{Ego, LiDAR}} \tag{3.3}$$

To obtain the relational transformation matrix between a source (S), i.e. the common coordinate frame, and a target (T), one can refer to (3.4). In this equation the first term represents the inverse of its global LiDAR relation.

$$T^{(\text{S, T})} = T^{(\text{S})}_{\text{LiDAR, Global}} \cdot T^{(\text{T})}_{\text{Global, LiDAR}} \tag{3.4}$$

$$T^{(\text{S})}_{\text{LiDAR, Global}} = T^{-1(\text{S})}_{\text{Global, LiDAR}} \tag{3.5}$$

Using the transformation matrix in (3.4) in (3.1) every sample's map elements can be defined in the common coordinate frame, thus we have recreated a scene from

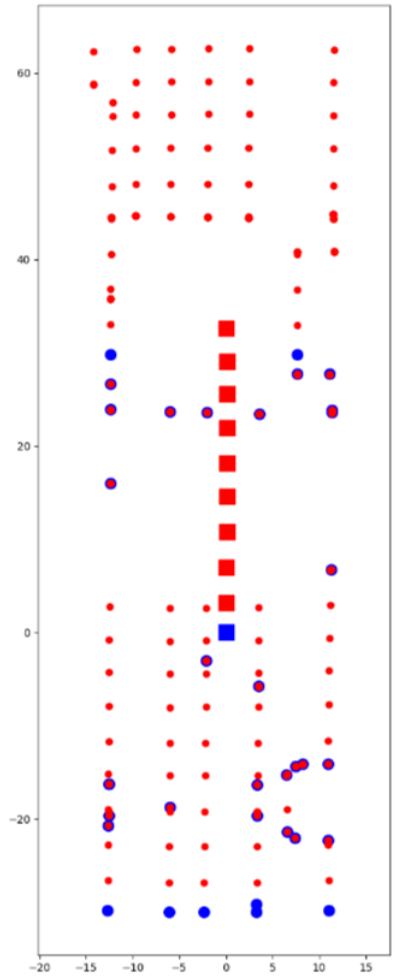


Figure 3.2: Simplified example of a subset of samples from an arbitrary scene in nuScenes. Blue is used as the common coordinate frame and the red is the subsequent samples in the subset. The dots correspond to map elements with no distinction between the classes, while the boxes represents the LiDAR sensor's position.

MapTRv2's predictions in a HD map format. A simplified example of how a subset of a scene looks after performing the coordinate transformation is depicted in Figure 3.2.

3.3.2 Rasterazation

Rasterazation refers to the process of converting a vector-based object into a raster format. In our case, this involves converting points into a grid of pixels or cells, which is a rather simple operation. The conversion is performed by dividing the point's coordinates (x, y) by a scalar representing the desired resolution w, resulting in the grid cell (u, v) shown in (3.6).

$$(u, v) = \frac{(x, y)}{w} \tag{3.6}$$

Currently, the representation of the map elements are polylines which are made up of point sets which now corresponds to a collection of grid cells. To get a more suitable representation of the polylines in the raster, the Bresenham's line algorithm is used. Bresenham's line algorithm is an efficient way to generate a straight line between two points in a grid-based system. The algorithm uses only integer addition, subtraction, and bit shifting, which makes it very fast. For details about the full implementation of Bresenham's line algorithm, see scikit-image's documentation of the function [25]. An illustration of Bresenham's algorithm can be seen in Figure 3.3.

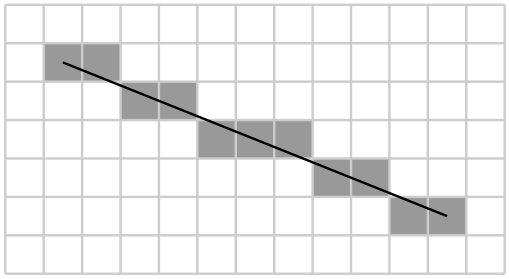


Figure 3.3: Illustration of the result of Bresenham's line algorithm.

3.4 Occupancy Grid Mapping

In this section we will explain how the class confidence provided by MatTRv2's classification branch can be utilized to increase the confidence in the rasterized pseudo scene maps with occupancy grid mapping.

Problem formulation:
As described in section 3.3 we create pseudo ground truths for each sample by building rasterized maps for scenes in nuScenes. Each of these maps are created/updated with occupancy grid mapping and the formulations we use are the following:

- m_i denotes grid cell i.
- x means the cell is occupied and $\bar{x}$ means the cell is empty
- z_t denotes a measurement at time t. Basically a sample at time t in the scene
- $l_{t,i}$ is the logs odds of a cell i being occupied at time t.

3.4.1 Grid Initialization

We initialize the grid by determining the smallest possible dimensions for each scene based on the BEV's extreme points of each sample, mentioned in 3.3.1. Each sample is defined by four corners, each with specific coordinates. To establish the grid dimensions, we identify the minimum and maximum x- and y-components respectively across all samples. Using these extreme values, we calculate the dimensions of the grid: the smallest and largest x-components determine the width, while the smallest and largest y-components determine the height. This ensures that the grid is appropriately sized to encompass the entire LiDAR range for all scenes, making it as compact as possible while fully covering the LiDAR range for every sample.

Following this, we equip the grid with three channels, one for each class label, an illustration of the grid structure can be seen in Figure 3.4. This design allows us to update of the grid independently for each class. Each channel is initialized with a prior specific to its class, set to a low value since most of a scene doesn't correspond to any of the map elements. The priors for lane dividers and boundaries are set slightly higher (0.04), than that for pedestrian crossings (0.02), as they typically occur more frequently.

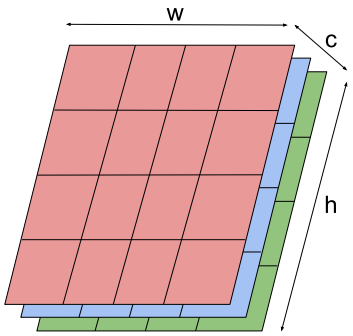


Figure 3.4: Illustration of a grid with three channels for each class, showing dimensions: width (w), height (h), and channels (c).

3.4.2 Inverse Sensor Model

In this project we use an inverse sensor model to update the map. It directly estimate the occupancy probability for each grid cell based on the sensor measurements. For example, $p(m_i = 1|z_t)$ denotes the probability of cell m_i being occupied given the sensor measurement z_t. The inverse sensor model also assumes that each cells state is independent of others, simplifying the computational load significantly.

This simplification contrast a forward sensor model, which need the entire map to be known or estimated and using a Maximum A Posterior (MAP) approach to identify the most likely map. Forward sensor models do not assume independence

among cells which can provide more accurate maps by resolving conflicting sensor measurements. However, this approach is very computational heavy. Therefore we choose the inverse sensor model based on its simplicity and efficient computation. These attributes are critical for quick updates and makes the inverse sensor model more suited for real-time applications.

3.4.3 Grid Update Algorithm

Before updating the grid, we apply Gaussian smoothing with a standard deviation of 3 to the rasterized predictions. This process ensures that cells further from the predicted occupied cell have a lower probability, forming a Gaussian distribution around the prediction. The advantage of Gaussian smoothing is that it softens the sharp boundaries of the predictions, distributing the occupancy probabilities.

These smoothed predictions are then used to update the occupancy grid according to our algorithm 1 which is based on Bayesian update mentioned in section 2.4.2. The algorithm specifies that for each sample in a sequence, the states of the cells with a prediction are updated, while the states of the other cells, i.e. those outside the perceptual field, remain unchanged.

The terms $p(\mathbf{m_i} = 1|z_{t-1})$ and $p(\mathbf{m_i} = 1|z_t)$ are the probabilities of cell m_i being occupied given measurement from previous sample z_{t-1} and the current sample z_t, respectively. The update of the cells inside the perceptual field involves three terms. The first and second terms represents the log odds of cell m_i being occupied or not, given z_{t-1} and z_t, respectively. The third term represent the prior knowledge of cell m_i being occupied or not. It adjusts for our initial assumption of the cells occupancy and it is independent of any sensor measurement. Figure 3.5 shows an example how a occupancy grid for a scene looks like after occupancy grid mapping with our algorithm using a pretrained MapTRv2.

Algorithm 1: Occupancy grid mapping

for *all cells* $\mathbf{m}_i$ **do**
  if $\mathbf{m}_i$ *has prediction* **then**
    $l_{t,i} = \log \frac{p(\mathbf{m_i}=1|z_{t-1})}{1-p(\mathbf{m_i}=1|z_{t-1})} + \log \frac{p(\mathbf{m_i}=1|z_t)}{1-p(\mathbf{m_i}=1|z_t)} - \log \frac{p(\mathbf{m_i}=1)}{1-p(\mathbf{m_i}=1)}$
  else
    $l_{t,i} = l_{t-1,i}$
  end
end

3.5 Supervision

One of the primary challenges of semi-supervised methods is determining how to effectively supervise them. As mentioned earlier, MapTRv2's BEV segmentation head, with modifications, can be utilized for this supervision. In this section, we

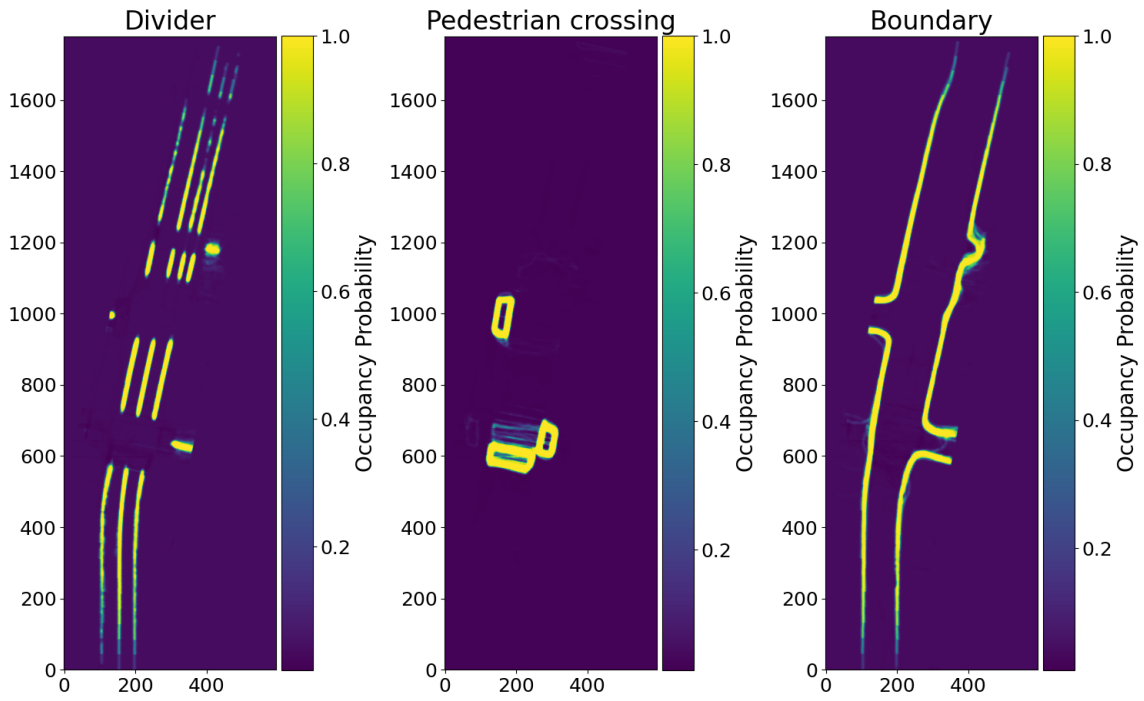


Figure 3.5: Example of a scene map created using occupancy grid mapping for an arbitrary scene in the nuScenes dataset. The map is generated from predictions by a pre-trained MapTRv2 model.

will show how the BEV segmentation head is adapted for semi-supervision, detailing the modifications and the formulation of the loss function.

3.5.1 Adaption of BEV Segmentation

In the original MapTRv2 model, the BEV segmentation for the auxiliary dense loss only includes foreground-background segmentation, effectively reducing all classes to a binary segmentation formulation. This approach limits the learning capabilities of the semantic classes. Extending this binary segmentation to semantic segmentation is straightforward. By segmenting three classes and splitting them into three separate BEV segmentation maps, the binary entropy loss between each channel of the occupancy grid and its corresponding BEV segmentation map defines the pseudo-loss, which is explained more in detail in the following subsection, which is then added to the overall loss of MapTRv2.

3.5.2 Pseudo-Loss

A critical aspect of supervision is the definition of the loss function. We propose a threshold-based strategy for defining the pseudo-loss. In this approach, cells with

a class probability greater than 90% are set to positive for that sample, and cells with a class probability less than 10% are set to negative. Probabilities between 10% and 90% are excluded from the loss computation. This strategy ensures that only highly confident predictions contribute to the loss calculation, thereby refining the learning process.

Threshold Strategy
Let $\hat{y}_c(i,j)$ be the predicted probability for cell (i,j) in channel c.

Define $f(p)$ as:

$$f(p) = \begin{cases} 1 & \text{if } p > 0.90 \\ 0 & \text{if } p < 0.10 \\ \text{undefined} & \text{if } 0.10 \leq p \leq 0.90 \end{cases} \tag{3.7}$$

Define the mask $m(p)$ as:

$$m(p) = \begin{cases} 1 & \text{if } p > 0.90 \text{ or } p < 0.10 \\ 0 & \text{if } 0.10 \leq p \leq 0.90 \end{cases} \tag{3.8}$$

Pseudo-Loss Calculation
Let $y_c(i,j)$ be the pseudo ground truth for cell (i,j) in channel c. The binary cross-entropy loss for cell (i,j) in channel c is given by:

$$\text{BCE}(\hat{y}_c(i,j), y_c(i,j)) \tag{3.9}$$

The pseudo-loss $\mathcal{L}_{\text{pseudo},c}$ for a single channel c can be expressed as:

$$\mathcal{L}_{\text{pseudo},c} = \sum_{i,j} m(\hat{y}_c(i,j)) \cdot \text{BCE}(f(\hat{y}_c(i,j)), y_c(i,j)) \tag{3.10}$$

Total Pseudo-Loss
The total pseudo-loss $\mathcal{L}_{\text{total}}$ is the sum of the pseudo-losses for all channels:

$$\mathcal{L}_{\text{pseudo,total}} = \sum_{c=1}^{3} \mathcal{L}_{\text{pseudo},c} \tag{3.11}$$

Combining with Overall Loss
Let $\mathcal{L}_{\text{MapTRv2}}$ be the original loss of the MapTRv2 model. The combined loss $\mathcal{L}$ can be defined as:

$$\mathcal{L} = \mathcal{L}_{\text{MapTRv2}} + \lambda \mathcal{L}_{\text{pseduo,total}} \tag{3.12}$$

where λ is a weighting factor for the pseudo-loss.

3.5.3 Dynamic Weighting

In the early stages of training, the model is expected to produce lower-quality predictions, as is common with most untrained models. This issue is partially mitigated by the threshold strategy, which requires cells to have high confidence to be included. Since each additional cell included in the loss calculation increases the loss

unless its contribution is zero, the model is effectively forced to reduce the contribution of these cells to zero in order to prevent the loss from deteriorating. To further address this, we regulate the model's reliance on the pseudo ground truth as training progresses. To achieve this effectively, we implement dynamic weighting. Initially, we set a lower weighting factor (λ), and as the training progresses and the model improves, (λ) increases linearly over the training epochs until it reaches a predetermined upper limit of 4.

3.6 Training Loop

The training process involves multiple epochs. During each epoch, the model first makes predictions on the entire dataset. These predictions are then used to construct/update scene maps for all unique scenes to which the unlabeled samples belong.

Following this, the model iterates through the samples by drawing a batch of both labeled and unlabeled samples. The labeled samples are processed through the MapTRv2 training pipeline as usual, while the unlabeled samples are processed only in the segmentation head. The segmented unlabeled samples are then compared to the pseudo ground truth (scene maps) in a binary cross entropy loss described in section 3.5.2. This pseudo loss is then added to the total loss which is determined as the sum of all the losses generated from the labeled samples.

3.7 Experimental Setup

We train Semi-MapTR on 4 GPUs with a batch size of 24, consisting of 16 labeled samples and 8 unlabeled samples. These are distributed evenly among the 4 GPUs, with each GPU receiving 4 labeled samples and 2 unlabeled sample. This ultimately means that the model will see the same labeled data several times within the same epoch. For the image backbone, we use a pre-trained ResNet50 [26]. For the LiDAR backbone, we use SECOND [27], trained from scratch. Both backbones are implemented as proposed by the authors of MapTRv2.

3.8 Evaluation Metrics

To evaluate the performance of the model we follow the same metric used by MapTRv2, i.e. Chamfer distance. Chamfer distance is a metric which is used to evalute the similarity between two sets of points, which is very convenient when working with polylines. Formally, given two point sets A and B, the Chamfer distance ($D_{Chamfer}(A, B)$) is calculated as follows: for each point in set A, determine the distance to its nearest neighbor in set B and sum these distances. Likewise, for each point in set B, find the distance to its nearest neighbor in set A and sum these distances. The Chamfer distance is the total of these two sums.

$$D_{Chamfer}(A, B) = \sum_{a \in A} \min_{b \in B} \|a - b\|^2 + \sum_{b \in B} \min_{a \in A} \|b - a\|^2 \tag{3.13}$$

Following MapTRv2, the average precision (AP) is calculated under several $D_{Chamfer}$ thresholds ($\tau \in T$, $T = \{0.5, 1.0, 1.5\}$). Finally, averaging across the thresholds the mean average precision (mAP) is obtained:

$$\text{AP} = \frac{1}{|T|} \sum_{\tau \in T} \text{AP}_\tau \tag{3.14}$$

We are also interested in evaluating whether MapTRv2's segmentation head improves with our contribution, and to measure this, we use Intersection over Union (IoU). IoU measures the overlap between the predicted segmentation mask and the ground truth mask. Formally, it is calculated as the area of intersection of the prediction (P) and the ground truth (GT) divided by the area of their union:

$$\text{IoU} = \frac{|P \cap GT|}{|P \cup GT|} \tag{3.15}$$

Then to get the mean IoU (mIoU) we average across all the segmented classes:

$$\text{mIoU} = \frac{1}{|\text{C}|} \sum_{\text{c} \in \text{C}} \text{IoU}_\text{c} \tag{3.16}$$

4
Results

In this chapter, we present a comprehensive summary of the experimental results. We begin by providing visualizations and detailed results of the map creation process, or pseudo GT, highlighting its evolution during model training and evaluating its quality. Following this, we compare the outcomes of training using only labeled data versus a combination of labeled and unlabeled data. Finally, we conduct an ablation study to analyze the contributions of various components of the model to its overall performance.

4.1 Map Creation

A crucial part of this project involves creating scene maps that serve as pseudo GT for semi-supervised training. In Figure 4.1-4.3 we can examine how a scene map evolves over time and what is included in the supervision for each class. Commonly, it is observed that in epoch 1 there's multiple predictions with low confidence in close vicinity of each other but as training progresses the results stabilizes. This is most notable for the pedestrian crossing class, where in Figure 4.2 the rasterized predictions are largely unsure and of poor quality initially but in epoch 24 we observe five nice quadratic-like predictions.

In the last plot of each Figure, we notice how the confidence threshold effects what should be included in the supervision. One can see that it accurately identifies the background and includes only those portions of the map elements for which there is a high level of confidence. This results in a clear outline around most of the map elements, effectively distinguishing between the map elements and the background. This is particularly beneficial as it ensures that the unconfident parts are not included in the supervision, thereby enhancing the quality of the training data.

Finally, Figure A.1 and Figure A.2 in Appendix 1, shows a comparison of pseudo GT, the mask, output from the segmentation head and GT for each map element. We recognize that the pseudo GT and GT corresponds very well, highlighting the success of our proposed method. Notably, the output from the segmentation head also corresponds pretty well with both the pseudo GT and GT, especially for the divider and boundary classes.

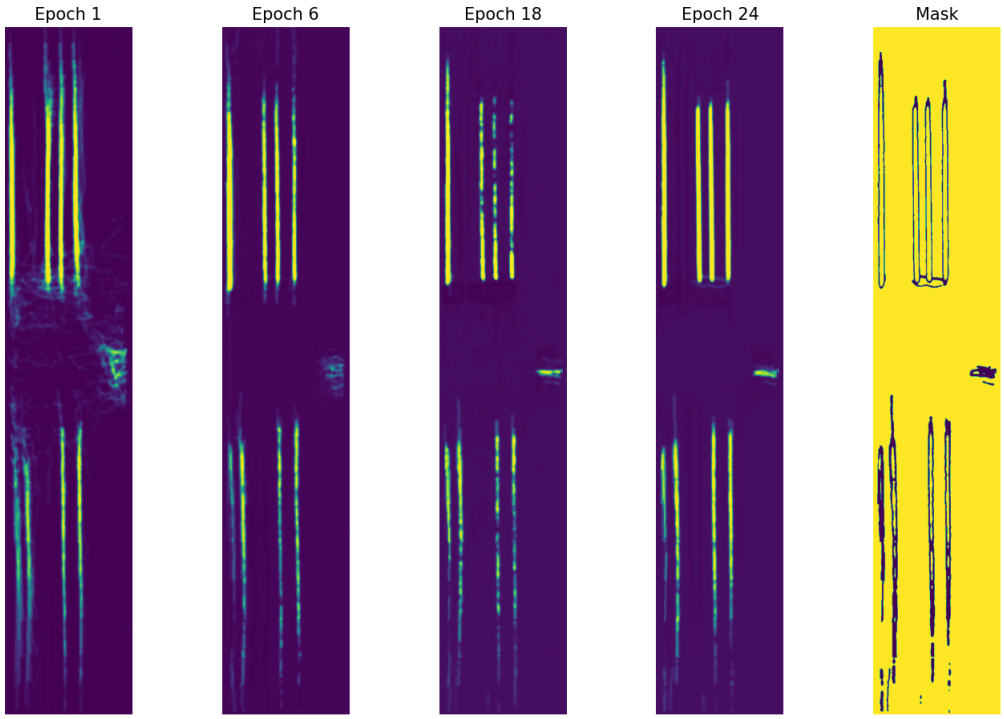


Figure 4.1: Evolution of a scene map for the divider class over epochs, accompanied by a mask showing the regions included in the loss function (yellow showing the positives of the mask).

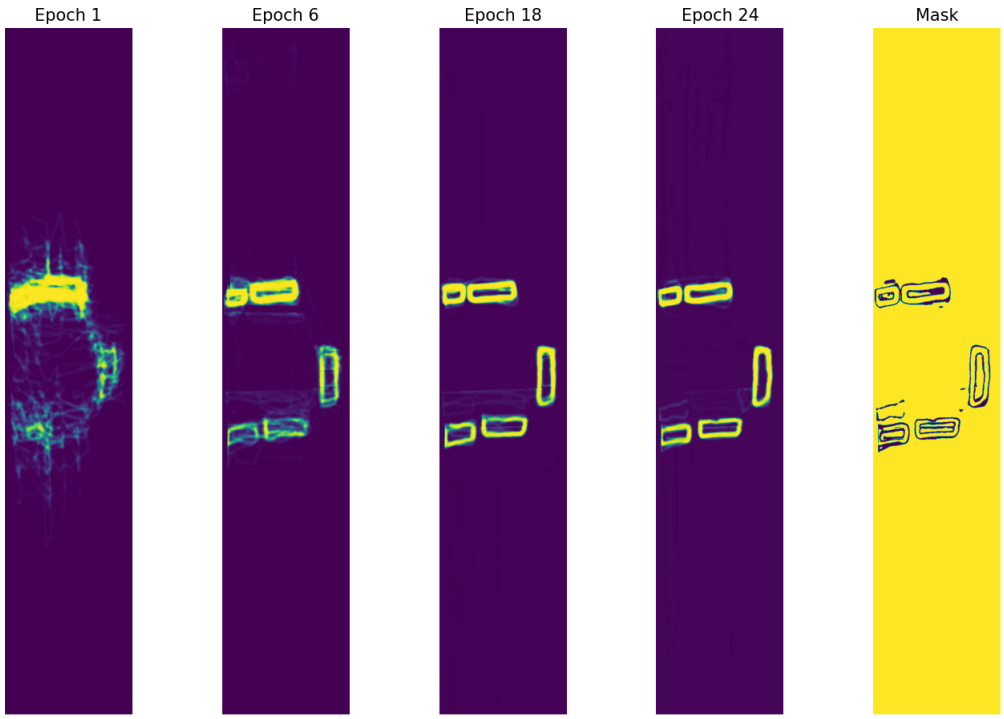


Figure 4.2: Evolution of a scene map for the pedestrian crossings class over epochs, accompanied by a mask showing the regions included in the loss function (yellow showing the positives of the mask).

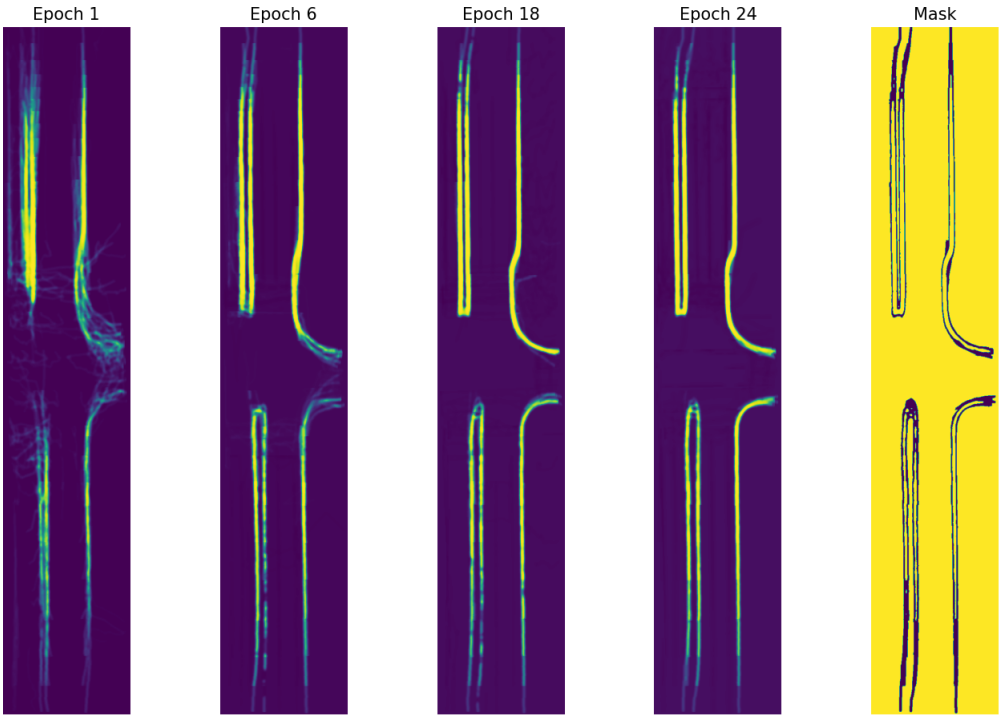


Figure 4.3: Evolution of a scene map for the boundary class over epochs, accompanied by a mask showing the regions included in the loss function (yellow showing the positives of the mask).

4.1.1 Evaluating the Pseudo GT

To determine if our model improves in generating pseudo GT, we leverage a unique opportunity rarely available in other scenarios: access to the GT for the unlabeled data. This allows us to directly compare the generated pseudo GT with the actual GT, using IoU. In Figure 4.4 it can be observed that as training progresses our model indeed becomes better at creating the scene maps. We see an equal performance for both camera and LiDAR data, but towards the end the maps created with camera data shows slightly better results.

It is also interesting to evaluate how different ratios of labeled and unlabeled data affect the model's ability to create pseudo GT. As demonstrated in Table 4.1 both modalities exhibit comparable results for each split, with the lowest performance occurring in the 10/90 split. The 50/50 split yields the highest mIoU, which is expected since more data is used for the supervised segment of the training. Interestingly, the decline in performance when reducing labeled data from 50% to 30% is not substantial, indicating that the model maintains robust performance even with less labeled data, at least to a certain point. This suggests promise for scalability, as it implies the model can still perform well with a reduced need for extensive labeled datasets.

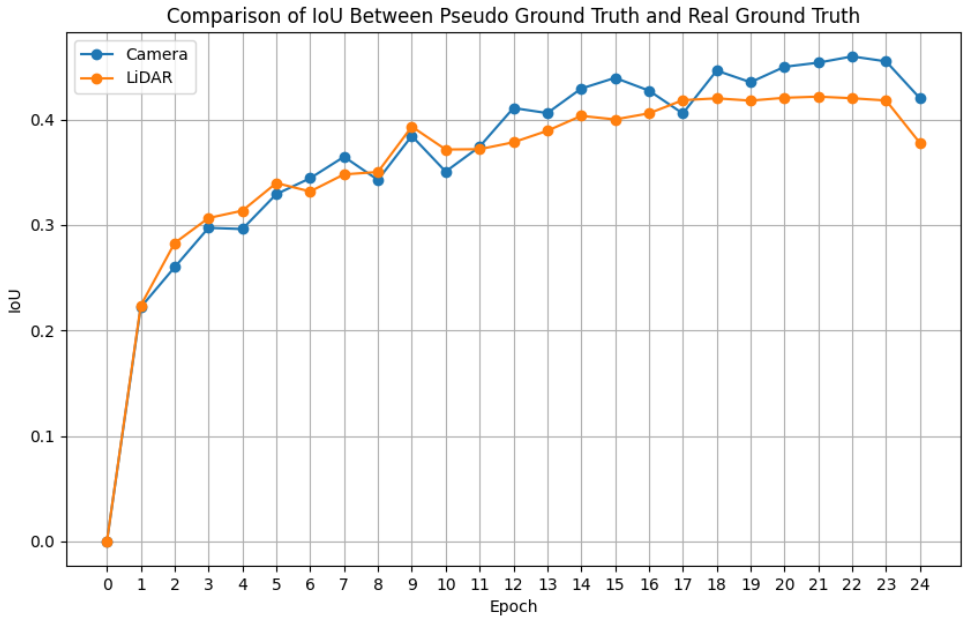


Figure 4.4: Comparison of IoU between the produced pseudo GT with the real GT correspondence during training on 30/70 split. The blue graph shows camera modality and the orange graph LiDAR modality. The IoU score is the average IoU calculated over all iterations within each epoch.

Table 4.1: Mean IoU performance between the produced pseudo GT with the real GT correspondence for different modalities and split ratios.

Modality	Split Ratio	mIoU
Camera	50/50	**0.50**
	30/70	0.47
	10/90	0.36
LiDAR	50/50	**0.49**
	30/70	0.42
	10/90	0.29

4.2 Evaluating Results of Semi-MapTR

Tables 4.2 and 4.3 compare the performances of MapTRv2 and Semi-MapTR across LiDAR and camera modalities, respectively. Both models were trained using the same amount of labeled data. Figures 4.5 and 4.6 display the IoU comparisons between MapTRv2 and Semi-MapTR at every second epoch, specifically for the 30/70 data split in both LiDAR and camera modalities.

From Table 4.2, it is evident that Semi-MapTR performance is best when trained on a split of 50% labeled and 50% unlabeled data both for LiDAR and camera data, which is to be expected as a larger portion of the training is supervised compared to the other splits. On the same amount of labeled data MapTRv2 demonstrates a inferior result compared to Semi-MapTR in every category, except for the divider class on camera data. This discrepancy is likely due to the fact that a larger amount of data, even though it is unannotated, has been used in training Semi-MapTR.

Additionally, for camera data, the performance doesn't drop significantly when reducing the amount of labeled data to 30%, with the AP for the divider class even increasing slightly. In contrast, for LiDAR data, we observe a significant drop in performance with the same reduction in labeled data. Another interesting observation is that Semi-MapTR consistently exhibits better results for the pedestrian crossing class across all splits when compared to MapTRv2.

Table 4.3 clearly demonstrates that implementing semi-supervision enhances segmentation, as mIoU increases across all categories. Conversely, Table 4.2 shows no significant improvements in segmentation. This suggests that the segmentation capabilities for camera data benefit more from semi-supervision compared to LiDAR data.

Another intriguing aspect to investigate is how segmentation improves over the epochs, as shown in Figure 4.5 and 4.6 for LiDAR and camera data, respectively. As anticipated from the results in Table 4.2, there are no significant differences in the segmentation improvement for LiDAR data when comparing MapTRv2 and Semi-MapTR. The notable observation is that Semi-MapTR improves more rapidly than MapTRv2. More interesting is the result shown in Figure 4.6 which demonstrates a vertical shift upwards for the Semi-MapTR curve compared to the MapTRv2 curve, indicating consistently better performance across all epochs.

Table 4.2: Performance comparison of Semi-MapTR with MapTRv2 on LiDAR data for different ratios of labeled and unlabeled data (L/U).

Method	Split Ratio	Epochs	AP				mIoU
	L/U		ped.	div.	bou.	mean	
MapTRv2	50/-	24	8.3	14.3	31.0	17.9	0.242
	30/-	24	6.3	13.1	28.6	16.0	0.238
	10/-	24	1.5	9.3	20.1	10.3	0.207
Semi-MapTR	50/50	24	**10.1**	**15.8**	**32.1**	**19.3**	**0.244**
	30/70	24	9.1	10.4	26.4	15.3	0.230
	10/90	24	1.6	5.0	17.0	7.9	0.211

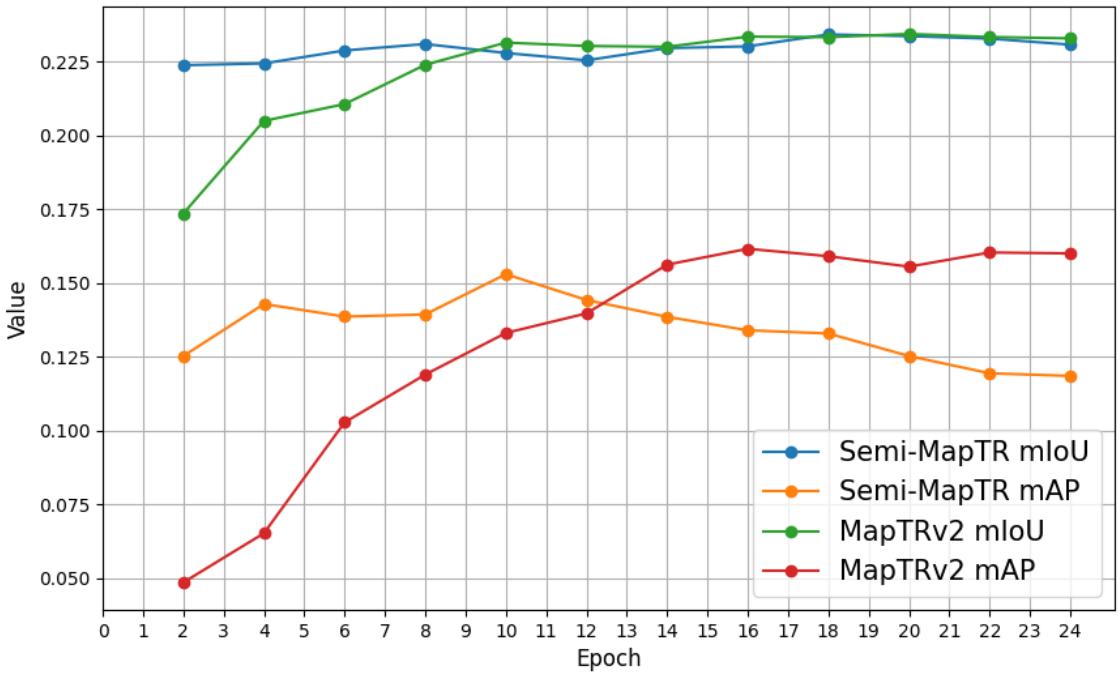


Figure 4.5: mAP and mIoU result on validation dataset for every second epoch. Comparing Semi-MapTR with MapTRv2, for 30/70 L/U with LiDAR data.

Table 4.3: Performance comparison of Semi-MapTR with MapTRv2 on camera data for different ratios of labeled and unlabeled data (L/U).

Method	Split Ratio	Epochs	AP				mIoU
	L/U		ped.	div.	bou.	mean	
MapTRv2	50/-	24	19.8	16.2	27.9	21.3	0.256
	30/-	24	13.0	13.8	24.5	17.1	0.242
	10/-	24	6.4	10.0	17.9	11.5	0.216
Semi-MapTR	50/50	24	**23.3**	15.9	**29.5**	**22.9**	**0.283**
	30/70	24	18.5	**16.6**	28.9	21.3	0.268
	10/90	24	12.6	7.9	21.0	13.8	0.237

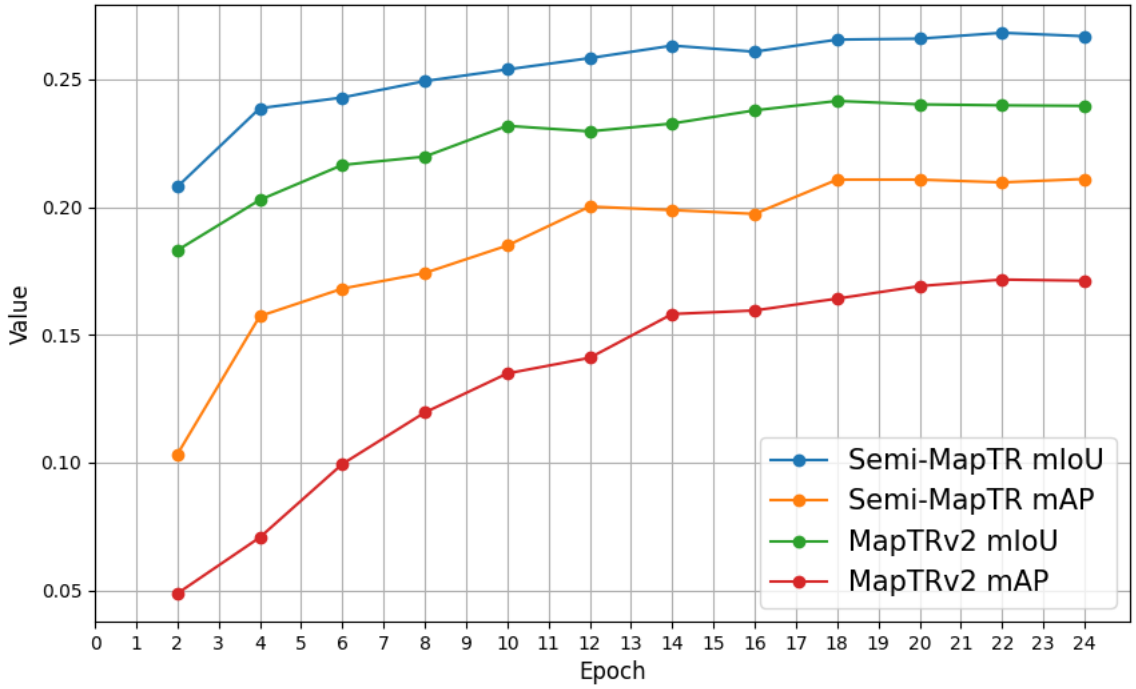


Figure 4.6: mAP and mIoU result on validation dataset for every second epoch. Comparing Semi-MapTR with MapTRv2, for 30/70 L/U split with camera data.

4.3 Ablation Studies

In this section, we present the results of the ablation studies aimed at identifying the most important parts of the model and understanding how they contribute to the model's performance. Each experiment is conducted using the same configuration: a 30/70 split of labeled and unlabeled data.

4.3.1 Different Confidence Thresholds

Table 4.4 shows the ablation for the upper confidence threshold (τ) when creating the mask for the pseudo GT. We observe a trend that having a stricter τ is most

beneficial for the model, especially when looking at mAP for LiDAR data. This improvement can be attributed to the enhanced ability to recognize the pedestrian crossing class as τ is increased from 60% to 90%. Interestingly, there is no notable effect on segmentation performance across different thresholds.

Examining the results for the camera data reveals no clear trends, although there is a slight improvement in performance at $\tau = 70\%$. However, it is evident that a 60% threshold is the least advantageous.

Table 4.4: Ablation on upper mask threshold (τ) for LiDAR and camera data.

LiDAR						**Camera**					
τ	**AP**				**mIoU**	τ	**AP**				**mIoU**
	ped.	**div.**	**bou.**	**mean**			**ped.**	**div.**	**bou.**	**mean**	
90%	**9.1**	10.2	26.4	**15.2**	0.234	90%	18.5	16.6	**28.9**	21.3	0.274
80%	7.1	9.5	24.9	13.8	0.233	80%	19.5	15.2	26.8	20.5	**0.275**
70%	4.9	10.8	**28.2**	14.6	**0.237**	70%	**21.7**	**17.0**	28.5	**22.4**	0.274
60%	5.1	**12.9**	25.9	14.6	0.234	60%	17.1	15.5	28.8	20.4	0.273

4.3.2 Dynamic Weighting

To assess the impact of dynamic weighting (DW), we compared it against both static weights and no weights (i.e. $w = 1$), using camera data. The results in Table 4.5 indicate that dynamic weighting does not offer significant advantages over static weighting. While there is a slight improvement in AP for boundary detection, performance for pedestrian crossing deteriorates. Notably, using no weights at all appears to yield better results, with superior performance for pedestrian crossing, thereby boosting the mAP by 0.2. Overall, there is minimal difference between the weighting approaches, but it seems that higher weights benefit the divider class, whereas no weight benefits pedestrian crossing.

Table 4.5: Ablation study of weight balancing (w). Comparison of different weightings.

w	**AP**				**mIoU**
	ped.	**div.**	**bou.**	**mean**	
0.5	19.4	15.3	27.6	20.8	0.263
1	22.0	15.4	27.2	21.5	0.265
2	20.1	16.6	27.3	21.3	0.271
4	20.0	16.3	27.6	21.3	0.271
DW	18.5	16.6	28.9	21.3	0.268

4.3.3 Closing the Gap

In this subsection we compare MapTRv2 trained supervised, Semi-MapTR and MapTRv2 trained supervised but adding more labeled samples to the segmentation head.

More specifically, we evaluate the different training scenarios in table 4.6

Table 4.6: Training Scenarios and Descriptions

Training Scenario	Description
MapTRv2-50	Trained on 50% labeled data
Semi-MapTR-50/50	Trained with 50% labeled and 50% unlabeled data
MapTRv2-50+100Seg	Segmentation head trained on 100% labeled data, and the rest on 50% labeled data
MapTRv2-100	Trained on 100% labeled data
MapTRv2-10	Trained on 10% labeled data
Semi-MapTR-10/90	Trained with 10% labeled and 90% unlabeled data
MapTRv2-10+100Seg	Segmentation head trained on 100% labeled data, and the rest on 10% labeled data

Figure 4.7 illustrates that MapTRv2-50 shows lower precision across all classes comparing to the other configurations. This was expected since less labeled data usually limits the models ability to learn generalized features. However, the model gets better when adding more samples to the segmentation head, both when adding labeled samples (MapTRv2-50+100Seg) and unlabeled samples (Semi-MapTR-50/50).

Notably, Semi-MapTR-50/50 is very close in performance to the 100% supervised training of MapTRv2. This indicates that the performance gap between semi-supervised learning using 50% labeled and unlabeled data, and fully supervised learning is not so significant. Furthermore, Semi-MapTR-50/50 is even better than mapTRv2-50+100Seg.

In figure 4.8, Semi-MapTR-10/90 appear to close the gap between MapTRv2-10 and MapTRv2-10+100Seg. It also dubble the performance in AP for the pedestrian crossing class comparing to MapTRv2-10. On the other hand adding samples to the segmentation head reduces performance for the divider class, where MapTRv2-10 performs better. This behavior may stem from the fact that the GT isn't always adept at distinguishing between a boundary and a divider. In many cases, what is defined as a boundary in the GT would be more accurately described as a divider. This is potentially something that is effectively captured by the pseudo GT, which could explain the poorer results.

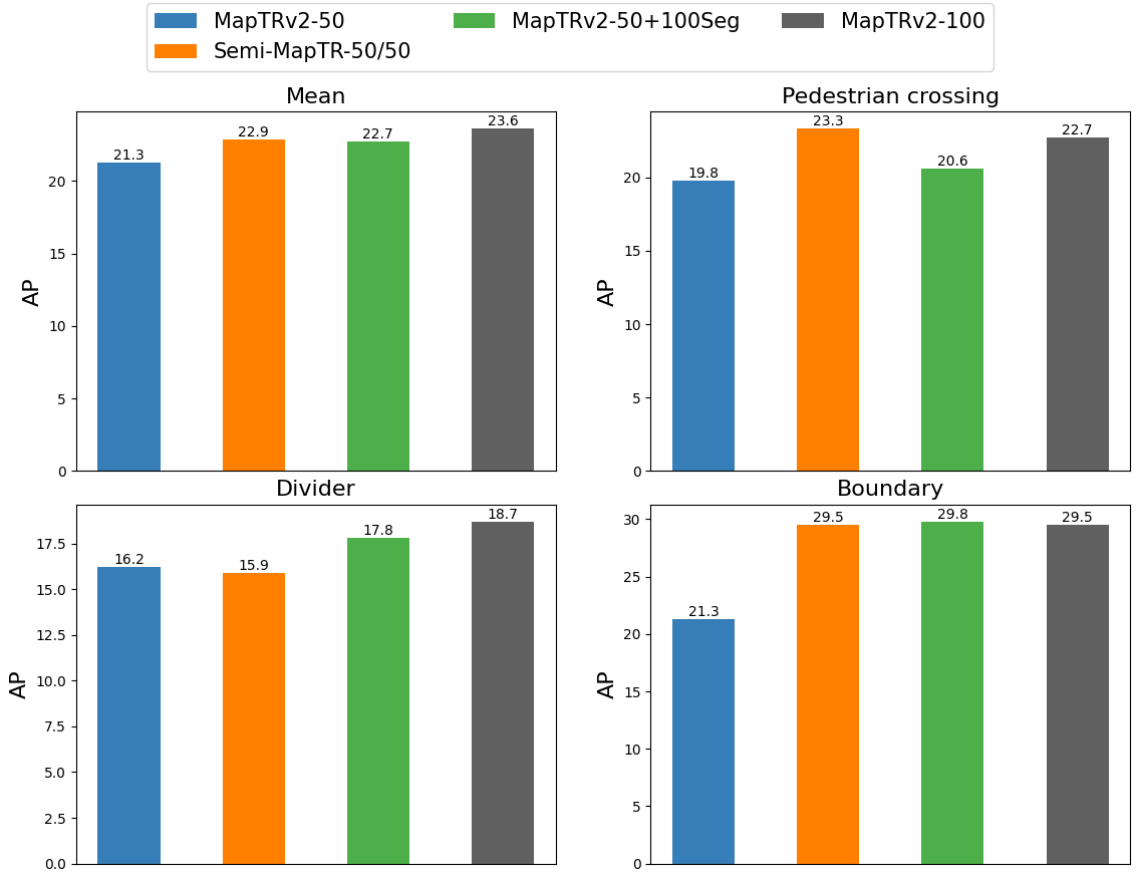


Figure 4.7: Bar chart comparing AP for the training scenarios; MapTRv2-50, MapTRv2-50+100Seg, MapTRv2-100 and Semi-MapTR-50/50.

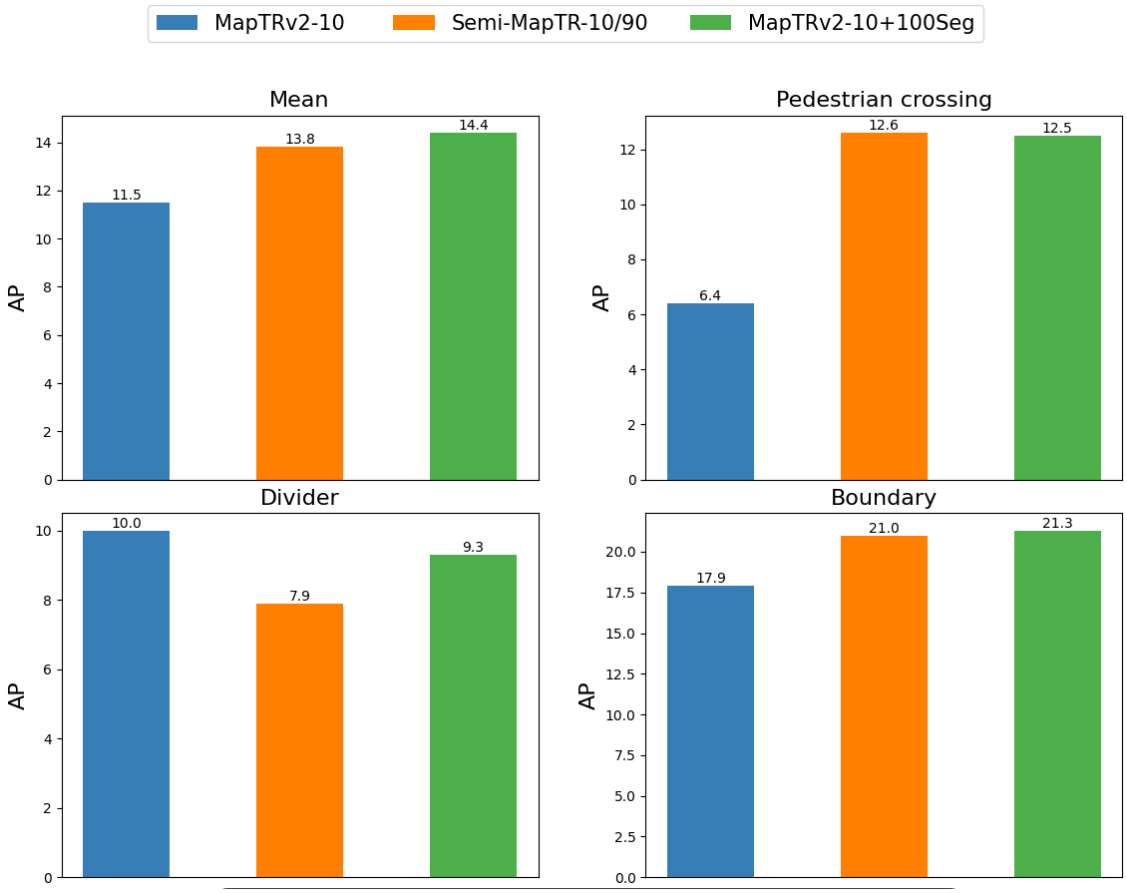


Figure 4.8: Bar chart comparing AP for the training scenarios; MapTRv2-10, MapTRv2-10/90, MapTRv2-10+100Seg.

5
Discussion

In this chapter, we will analyze the results presented in the previous chapter, starting with a discussion of how good the pseudo GT are in comparison to the real GT. Furthermore, the effects of adding semi-supervision to MapTRv2 is discussed as well as the differences between the two modality data and finally the results of the ablation studies.

5.1 Map Creation

As previously mentioned, an essential part of this project is the creation of high-quality pseudo GT. In Section 4.1, we observed that the pseudo GT aligns closely with the actual GT for both LiDAR and camera data, and that the scenes improve progressively over the epochs. We also noted that the segmentation head's output generally matched both GTs well, although the pedestrian crossing class displayed a more clustered appearance. Noticeable, in Figure A.2 and A.1 reveal that the segmentation head struggles more with segmenting shorter map elements compared to longer, coherent map elements. This is logical, as longer map elements would appear more frequently across samples, especially when the dataset has been downsampled to 1 Hz. This affects the model in two ways: it increases the model's exposure to those particular map elements, and it enhances the occupancy grid mapping's accuracy when estimating confidence for map elements during the map creation step. Overall, this approach has shown great promise in creating an accurate and effective representation for scene maps in a reliable manner.

5.2 Adding Semi-Supervision to MapTRv2

From the result part we saw that adding semi-supervision to MapTRv2 improves the models precision in every category except for the divider class, and notably improves the mIoU for the camera modality. This highlights the benefits of adding unlabeled data into the segmentation head, leading to a more generalized segmentation head and, consequently, a more generalized overall model.

The effect of semi-supervision was not as promising for LiDAR data, but as mentioned in the result part it was still performing better on the pedestrian crossing class, which is very intriguing. This suggest that adding semi-supervision, by generating pseudo GT, enhances the models ability to predict minority classes, also with more complex structure.

Minority Classes like pedestrian crossing, may require more data to be captured, also their complex structure such as varied shapes and intricacies may require more data. The use of unlabeled data with pseudo GT for the segmentation head can help providing this extra data, and thereby improving the model performance for these not so frequently occurring classes. Furthermore, using pseudo GT, which are not as perfect as the real GT can add some variability and further help the model to generalize better to the minority classes and perhaps classes with more complex shapes.

The above mentioned phenomenon can be visualized in the Figures 4.7 and 4.8 where the pedestrian crossing class gain in performance using pseudo GT. These figures also in combination with the tables in Section 4.2 shows that occupancy grid mapping is an effective method for creating pseudo GT in semi-supervision.

5.2.1 Impact of Data Modalities

From the results in Table 4.2 and Table 4.3 it can be observed that Semi-MapTR benefits more from camera data than it does from LiDAR data. This could be because LiDAR measure distances and reflectivity, which is crucial for understanding object positions and shapes but it lacks the visual features that the camera provides. Consequently this can make the addition of unlabeled samples for LiDAR modality less beneficial comparing to the camera modality.

The absence of fine-tuning for each specific modality adds uncertainties in the comparison between the camera modality and LiDAR modality. In this research we have used MapTRv2:s standard configuration hyperparameters such as learning rate and weight decay, which are optimized for the camera modality. These parameters might not be optimal for LiDAR data, potentially limiting the results and effect of adding semi-supervision to MapTRv2 for LiDAR data.

5.3 Confident Pseudo GT

The results from Section 4.3.1 showed a positive correlation between a stricter τ and improved mAP for LiDAR data suggesting that higher confidence in pseudo labels significantly enhances the model's ability to distinguish pedestrian crossings. This could be because LiDAR data, being more precise in spatial measurements, benefits more from higher quality pseudo labels. When τ is increased, the pseudo labels are more accurate, leading to better feature learning for classes that require precise localization, such as pedestrian crossings. This highlights the importance of having high quality pseudo labels, especially when in scenarios where spatial accuracy is important.

Both LiDAR and camera data shows that when $\tau = 60\%$ is the least advantageous, suggesting that this low of a threshold might be too lenient, allowing too many low-quality pseudo labels to influence the model. Being too lenient risks introducing noise and thus deteriorate the overall quality of the training data which hinders the learning process. This insight highlights the need for careful consideration of

confidence in the pseudo data.

In contrast to the results on LiDAR data, the camera data do not show a clear pattern, but a slight edge can be given to $\tau = 70\%$, which shows better performance for pedestrian crossing, boundary, and mAP. There's no apparent reason for this finding, but it might indicate that a moderate confidence threshold strikes a balance between including enough pseudo labels to provide diverse training data and maintaining label quality when using camera data.

5.4 The Effects of Dynamic Weighting

As can be observed in Table 4.5 having dynamic weighting, which weighs the pseudo-loss less in the early stage and progressively increasing it, has little impact showcasing comparable result to having a static weight. The preferable setting seems to be to have no weighting at all. This is likely because the threshold strategy alone sufficiently manages the reliance on pseudo GT. Since each additional cell included in the loss calculation increases the loss unless its contribution is zero, the model is naturally encouraged to reduce the contribution of low-confidence cells to zero. Therefore, the threshold strategy alone sufficiently manages the model's performance, rendering additional weighting strategies unnecessary.

6
Conclusion

In this chapter we will summarize the key findings of our research and reflect on their significance in relation to our research objectives. Following this we will discuss the limitations encountered during the project and from this discuss some future work possibilities.

The results demonstrate that Semi-MapTR outperforms MapTRv2 in terms of the precision of polyline predictions and IoU for segmentation when trained on the same amount of labeled data. An especially notable finding across all experimental runs is that the precision and IoU for the pedestrian crossing class consistently show better results with the semi-supervised approach. This suggests that SSL can be particularly beneficial for improving the precision and generalization of HD map construction, especially for less frequently occurring classes like pedestrian crossings. These findings effectively answer the first question about how SSL improves HD map construction compared to supervised methods, as Semi-MapTR achieves superior performance in almost every case in terms of both precision and IoU compared to MapTRv2.

This improvement is particularly notable with camera modality data compared to LiDAR modality data. In addressing the second research question about the impact of camera and LiDAR data on SSL versus supervised learning, our results indicate a more pronounced benefit of SSL for camera data. However, it is crucial to interpret these findings with caution. The current study did not fully explore hyper-parameter tuning and other optimization techniques, which could significantly influence the results. Consequently, while our findings are promising, further research is needed to validate and extend these conclusions.

6.1 Future Research

A key area for future research is the application of semi-supervision directly to polylines, rather than to segmentation. Since polylines represent the final output of the model, optimizing their loss function could have a more significant impact on the model's performance. approach presents considerable challenges, such as finding a efficient way to build the pseudo ground truth for it holds the potential to greatly enhance model generalization and precision.

Another potential avenue for future research involves comparing two different meth-

ods of splitting data into labeled and unlabeled sets: either by random samples or by random scenes. This comparison is relevant to two distinct real-world scenarios: one where the annotated data consists of random samples, and another where the annotated data comprises entire logs. This comparison could potentially help to determine which approach leads to better generalization on new unseen data by investigating how contextual information affects the model.

Finally, there are numerous tuning opportunities that could potentially enhance the model. Beyond conducting a comprehensive hyper-parameter scan for each labeled-unlabeled split ratio and experimenting with different combinations of learning rates and weight decays, another promising approach is to use temperature scaling for the logits of each cell in the occupancy grid.

By addressing these future research opportunities, the use of SSL for HD map construction could result in more accurate and generalized HD maps, significantly enhancing the scalability of online HD map construction.

Bibliography

[1] B. Liao, S. Chen, X. Wang, T. Cheng, Q. Zhang, W. Liu, and C. Huang, "Maptr: Structured modeling and learning for online vectorized hd map construction," 2023.

[2] Y. Liu, T. Yuan, Y. Wang, Y. Wang, and H. Zhao, "Vectormapnet: End-to-end vectorized hd map learning," 2023.

[3] B. Liao, S. Chen, Y. Zhang, B. Jiang, Q. Zhang, W. Liu, C. Huang, and X. Wang, "Maptrv2: An end-to-end framework for online vectorized hd map construction," 2023.

[4] T. Yuan, Y. Liu, Y. Wang, Y. Wang, and H. Zhao, "Streammapnet: Streaming mapping network for vectorized online hd map construction," 2023.

[5] Z. Xu, K.-Y. K. Wong, and H. Zhao, "Insmapper: Exploring inner-instance information for vectorized hd mapping," 2024.

[6] Z. Zhang, Y. Zhang, X. Ding, F. Jin, and X. Yue, "Online vectorized hd map construction using geometry," 2023.

[7] nuScenes, "nuscenes: A multimodal dataset for autonomous driving." `https://www.nuscenes.org/`, 2024. Accessed: 2024-05-17.

[8] Argoverse, "Argoverse 2: A dataset for autonomous vehicle perception and motion forecasting." `https://www.argoverse.org/`, 2024. Accessed: 2024-05-17.

[9] J. Zhang, X. Lin, W. Zhang, K. Wang, X. Tan, J. Han, E. Ding, J. Wang, and G. Li, "Semi-detr: Semi-supervised object detection with detection transformers," 2023.

[10] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," 2020.

[11] N. Gustaffsson and G. Rödström, "Semi-supervised learning for autonomous vehicle object detection," 2020.

[12] Q. Li, Y. Wang, Y. Wang, and H. Zhao, "Hdmapnet: An online hd map construction and evaluation framework," 2022.

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[14] M. Jiang, Y. Cheng, and L. Liu, "Mapseg: Segmentation guided structured model for online hd map construction," 2023.

[15] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, ACM, 1998.

[16] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, (Atlanta), p. 896, 2013.

[17] Z. Li, B. Ko, and H.-J. Choi, "Naive semi-supervised deep learning using pseudo-label," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 1358–1368, 2019.

[18] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," 2014.

[19] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," 2016.

[20] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," 2017.

[21] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," 2018.

[22] A. Elfes and L. Matthies, "Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representation," in *Decision and Control, 1987. Proceedings of the 26th IEEE Conference on*, pp. 1802–1807, IEEE, 1987.

[23] A. Styler and V. Hwang, "Statistical techniques in robotics (16-831, f12) lecture #05: Mapping," 2012. Lecture notes for Carnegie Mellon University. Available at `https://www.cs.cmu.edu/~16831-f12/notes/F12/16831_lecture05_vh.pdf`.

[24] A. Lilja, J. Fu, E. Stenborg, and L. Hammarstrand, "Localization is all you evaluate: Data leakage in online mapping datasets and how to fix it," 2023.

[25] scikit-image contributors, *skimage.draw.line*, 2024. Accessed: 2024-05-24.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

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

A

Appendix 1

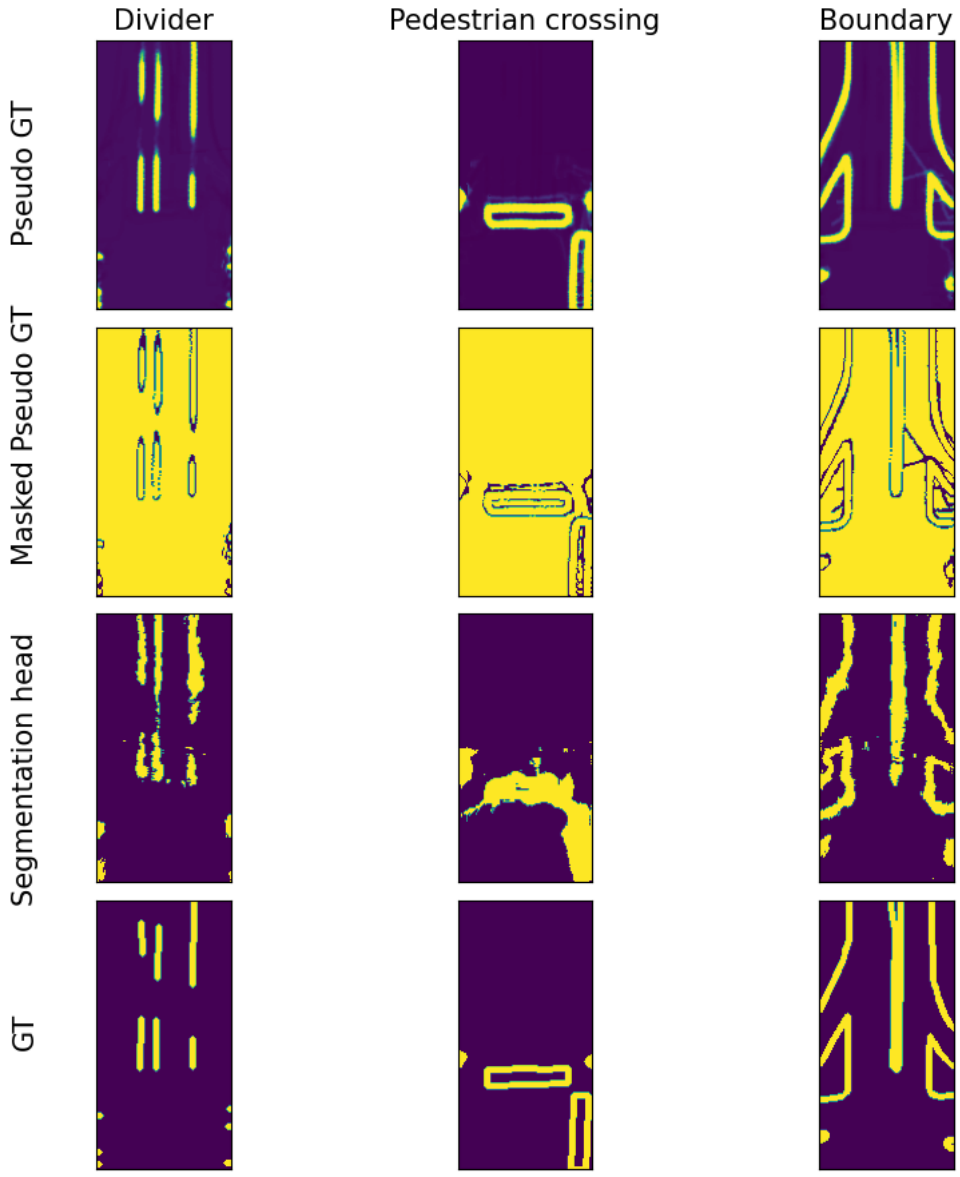


Figure A.1: Comparison of the pseudo GT, masked pseudo GT, output from segmentation head and the real GT for one unlabeled sample in the end of epoch 24. This is for the camera modality on the 30/70 split.

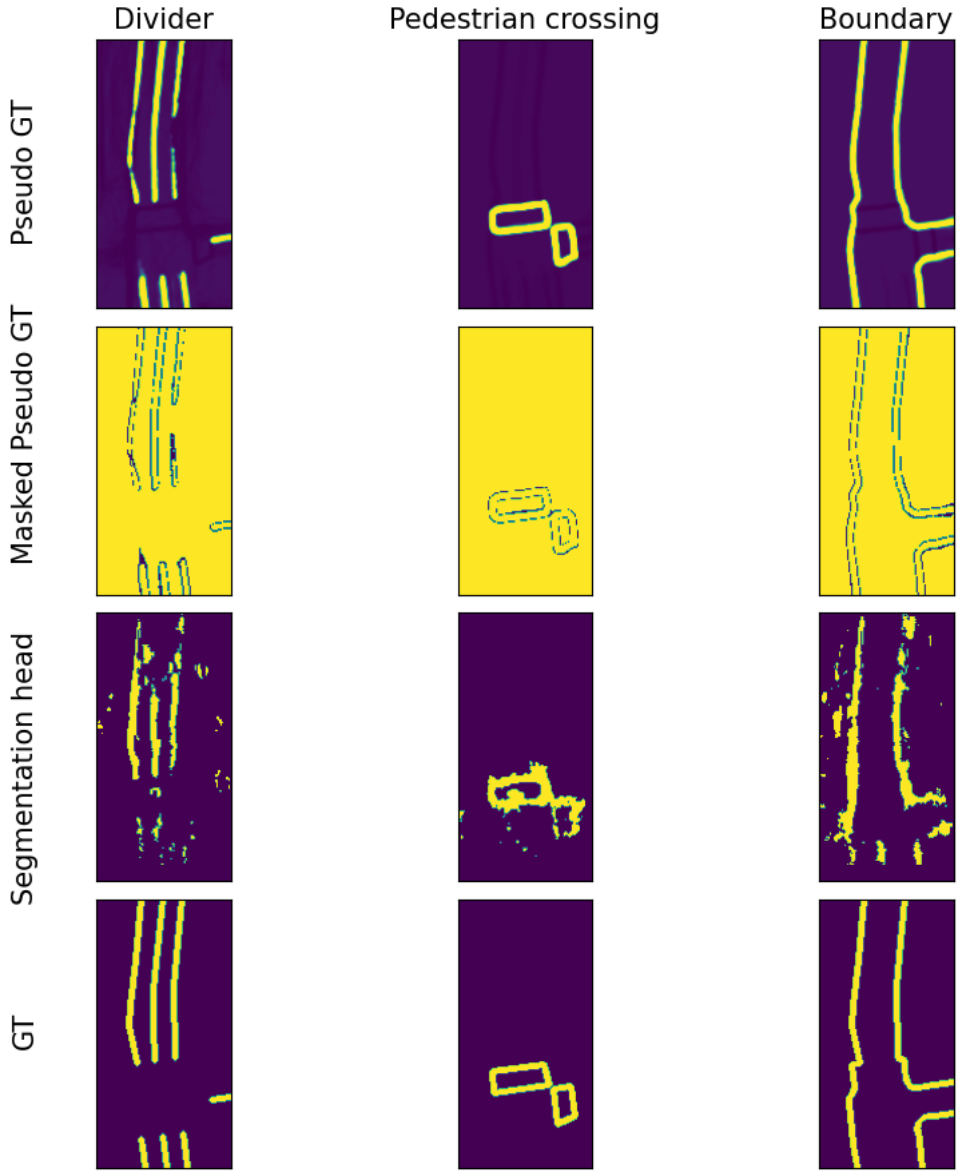


Figure A.2: Comparison of the pseudo GT, masked pseudo GT, output from segmentation head and the real GT for one unlabeled sample in the end of epoch 24. This is for the LiDAR modality 30/70 split.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY