Exploring Supervised, Semi-supervised and Self-supervised Learning in Autonomous Driving

Partha Ghosh
University of Tübingen

Abstract

How can we learn generalized autonomous driving models for robust vision-based navigation in complex and dynamic settings? The status quo for solving these visual navigation tasks is to train visual representations and navigation policies with direct supervision. Along with exploring some techinques to improve supervised driving performance, in this work, we mainly study several different approaches to effectively utilize vast amounts unlabeled, highly diverse ego-centric navigation data that is freely available on the internet to robustly scale across perspectives, platforms, environmental conditions, scenarios, and geographical locations. We study two existing works in this field that employes semi-supervised and self-supervised learning, namely 'SelfD' [69] and 'OVRL' [63]. Moreove, in this work, we introduce SemiD, a framework for learning scalable driving by utilizing large amounts of online monocular images. Our key idea is to leverage deep visual odometry for iterative semi-supervised training when learning imitative agents from unlabeled data. To handle unconstrained viewpoints, scenes, and camera parameters, we train an image-based model that directly learns to plan in the Bird's Eye View (BEV) space. We use deep visual odometry to generate pseduo-labels for the the unlabeled data and augment the decision-making knowledge and robustness of the model via semi-supervised training. We employ a large dataset of publicly available YouTube videos to train SemiD and comprehensively analyze its generalization benefits across challenging navigation scenarios. Without requiring any additional data collection or annotation efforts, SemiD outperforms all the previous approaches and demonstrates consistent improvements from 51% to 95% in route completion and from 7.8% to 13.3% in driving score in challenging CARLA evaluation routes.

Keywords: Imitation Learning, Self-supervised Learning, Semi-supervised Learning

Acknowledgements

I want to thank Bernhard Jaeger and Katrin Renz for supervising the project and providing me with the necessary guidance and valuable support. I want to thank Prof. Andreas Geiger for the helpful discussions. Finally, I want to express my very profound gratitude to my parents for their continuous support and encouragement throughout my whole life.

Contents

0.1 Introduction . . . 9
0.2 Related Work . . . 10
0.2.1 End-to-End Autonomous Driving . . . 10
0.2.2 Imitation Learning . . . 11
0.2.3 Semi-Supervised Learning for Navigation . . . 11
0.2.3.1 Semi-supervised Learning . . . 11
0.2.3.2 Learning by Cheating . . . 12
0.2.4 Self-Supervised Visual Representation Learning . . . 12
0.2.4.1 Self-supervised learning . . . 12
0.2.4.2 Contrastive Learning . . . 12
0.2.4.3 Non-Contrastive Learning . . . 13
0.2.4.4 Action-COnditioned Policy Pretraining (ACO) . . . 13
0.3 Autonomous Driving Framework . . . 13
0.3.1 Imitation Learning . . . 14
0.3.2 Input and Output Representations . . . 14
0.3.2.1 Input Representation . . . 14
0.3.2.1.1 Driving Scenes . . . 14
0.3.2.1.2 Global Planner . . . 16
0.3.2.2 Output Representation . . . 16
0.3.3 Waypoint Prediction Network . . . 16
0.3.4 PID Controller . . . 17
0.3.5 Loss Function . . . 17
0.4 Learning Approaches . . . 17
0.4.1 SemiD: Semi-supervised Driving with Deep Visual Odometry . . . 18
0.4.1.1 Deep Visual Odometry . . . 18
0.4.1.1.1 Feature-encoding Module . . . 18
0.4.1.1.2 Memory-propagating Module . . . 19
0.4.1.2 Generating Pseduo-labels for Video frames . . . 20
0.4.1.3 Model Pre-Training and Fine-Tuning . . . 21
0.4.2 SelfD: Self-Learning Large-Scale Driving Policies . . . 21

0.4.2.1 Conditional Imitation Learning from Observations 23
0.4.2.2 Initial Data Assumption 23
0.4.2.3 Self-supervised Training Process 23
0.4.2.4 BEV Plan Network 23
0.4.2.5 “What If” Pseudo-Labeling of the Unlabeled Data 24
0.4.2.6 Model Pre-Training and Fine-Tuning 25
0.4.3 OVRL: Offline Visual Representation Learning 25
0.4.3.1 Self-supervised Pretraining 25
0.4.3.2 Implementation Details 27
0.4.3.3 Downstream Learning 27
0.5 Experimental Results 27
0.5.1 Task 28
0.5.2 Implementation Details 29
0.5.2.1 Data Cleansing 29
0.5.2.2 Stratified Sampling 29
0.5.2.3 Image Augmentation 29
0.5.3 Dataset 29
0.5.4 Evaluation Metrics 30
0.5.4.1 Route Completion (RC) 30
0.5.4.2 Infraction Score (IS) 30
0.5.4.3 Driving Score (DS) 31
0.5.5 Results 31
0.5.5.1 Performance of Supervised Training 31
0.5.5.2 Perfomance of the Learning Approaches 32
0.6 Conclusion 36

0.1 Introduction

How should we teach autonomous systems to drive based on visual input? How can we learn generalized models for robust vision-based navigation in complex and dynamic settings? While humans can effortlessly transfer general navigation knowledge across settings and platforms (e.g. geographical location, use-case, rare-scenarios, camera mounting point), current navigation agents cannot transfer this knowledge well. With this question in mind, we are therefore, interested in exploring learning strategy with which navigation agents can learn to understand, across all settings and platforms, the structure and semantics of their environments and navigate accordingly without providing extensive direct supervision.

The status quo for solving these visual navigation tasks is to train visual representations and navigation policies from scratch with direct supervision. The family of approaches that has demonstrated promising results is imitation learning [15, 21]. The agent is given trajectories generated by an expert driver, along with the expert's sensory input. The goal of learning is to produce a policy that will mimic the expert's actions given corresponding input [1, 6, 13, 14, 29, 31, 38, 39, 44, 36, 46, 47, 48, 68]. Now, every minute, vast amount of highly diverse and freely available ego-centric navigation data containing such scenarios are uploaded to the web. Even though the expert's actions may not be readily available from these demonstration data, these data can be parsed to recover the corresponding expert's action. Another feasible approach could be to learn better representations from these unlabeled data which is a very popular topic in visual recognition. The learned representations are shown to be generalizable across visual tasks ranging from image classification, semantic segmentation, to object detection [16, 28, 32, 10, 9]. However, these methods are primarily built for learning features for recognition tasks rather than navigation. Therefore, in this work we aim towards effectively utilizing such freely available demonstration data to improve the efficiency, safety, and scalability of generalized real-world navigation agents.

We explore two different type of learning in the context of self-driving that facilitates learning from large amounts of unlabeled experience (combined with a small amount of direct supervision): (1) Semi-supervised Learning, (2) Self-supervised Representation Learning. Both semi-supervised and self-supervised methods are similar in the sense that they both facilitate learning from large amounts of unlabeled experience, but the way both formulate this, is quite differently. In semi-supervised learning, we devise strategies to generate reasonable pseduo-labels to the unlabeled input driving scenes so that upon training with these pseduo-labels the network can learn better generalized representations of these diverse driving scenes and therefore perform better in the navigation task. On the other hand, in self-supervised learning through different approaches (e.g. contrastive learning, entropy regulation) the network directly learns good representation of the unlabeled inputs. We also try to combine both of these approaches.

In this thesis, we aim to build and compare three different learning approaches — two of

which can be catogorized as semi-supervised learning and the other one as self-supervised learning. We incorporate and explore an existing semi-supervised learning approach "SelfD" proposed by Zhang et. al. [69] in our autonomous driving framework. We explore in our self-driving framework, "OVRL", a self-supervised learning approach proposed by Yadav et. al. [63] which has been studied in the domain of embodied navigation. Moreover, we propose SemiD, a new semi-supervised learning method that outperforms the prior works in the challenging NEAT evaluation routes [18]. Also, we propose a data cleansing and sampling technique for effective training. We show that combining the aforementioned technique with mild image augmentation improve the driving performance by a huge margin.

In summary, the main contributions of this work are:

- We present SemiD, a novel semi-supervised learning approach based on deep visual odometry for autonomous driving that outperforms prior works in the challenging NEAT evaluation routes.
- We showed that combining SemiD and OVRL brings further improvements in route completion.
- We propose a data cleaning pipeline and stratified sampling in training to improve the driving perfomance.
- We find that image augmentations is quite important for achieving good performance.
- We show that the inertia problem can be solved by employing the above techniques.

We organize the structure of the thesis as follows. We first provide an overview of the related works in this field in Section 0.2. Then, in Section 0.3, we introduce our autonomous driving framework where we will deploy all our learning approaches. In Section 0.4, we describe in details various semi-supervised and self-supervised learning approaches. Next, we discuss the experiment results in Section 0.5, followed by the conclusion in Section 0.6.

0.2 Related Work

0.2.1 End-to-End Autonomous Driving

End-to-End driving describes approaches in which the entire driving task is done by a single neural network that directly maps the raw sensor data to the driving commands. The neural network can be trained using different algorithms, the two most important ones being imitation learning and reinforcement learning. Even though the models are hard to interpret, the advantages of this approach is that these models can be optimized directly for driving. Furthermore, data annotations are cheap, since a camera can be attached to a car and sensors to the steering mechanisms to collect data automatically. This approach was used early on by researchers like Pomerleau et al. and their ALVINN-vehicle [47] and still active research

is going on in this field [33, 52]. Imitation Learning for driving has advanced significantly [6, 21, 44] and is currently employed in several state-of-the-art approaches, some of which predict waypoints [11, 15, 24], whereas others directly predict vehicular control [4, 7, 22, 45, 62, 49]. While other learning-based driving methods such as affordances [51, 61] and reinforcement learning [14, 55, 57] could also benefit from a semi-supervised or self-supervised learning, in this work, we try to improve imitation learning based autonomous driving through semi/self-supervised learning.

0.2.2 Imitation Learning

Imitation Learning is the most promising approach for self-driving. Our main idea is to leverage the scale and diversity of readily accessiblem online ego-centric navigation data to learn a robust conditional imitation learning policy [15, 21]. While learning from labeled demonstrations can significantly simplify the challenging vision-based policy learning task [1, 6, 13, 14, 29, 31, 38, 39, 44, 36, 46, 47, 48, 68, 71, 72], observed images in our settings are not labeled with corresponding actions of a demonstrator. Therefore we aim to generalize current conditional imitation learning (CIL) approaches [15, 21, 22] to learn, from unlabeled image observations, an agent that can navigate in complex urban scenarios. To address this challenging observational learning task, prior work has recently explored introducing various restrictive assumptions, including access to a hand-designed reward function [12], an interactive environment for on-policy data collection [53], or demonstrator optimality [53, 54]. We instead facilitate scalable training from diverse data sources, by employing semi-supervised or self-supervised learning approaches. Our resulting model can also be used to bootstrap other methods for policy training, e.g., model-based or model-free reinforcement learning approaches [14, 39, 45, 55].

0.2.3 Semi-Supervised Learning for Navigation

0.2.3.1 Semi-supervised Learning

Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. It falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). It is a special instance of weak supervision. Semi-supervised learning more closely imitates the way humans learn. In semi-supervised learning, the neural network learns in two steps:

- *Transductive Learning:* First, the task is solved based on pseudo-labels (i.e. by labelling the given unlabelled data) which help to initialize the network weights.
- *Inductive Learning:* The pre-trained network is then fine-tuned with the small ground truth data.

0.2.3.2 Learning by Cheating

The prior works [8, 64, 37, 50], emphasizing semi-supervised learning through image and object-level recognition tasks, has limited utility for complex decision-making tasks. On the other hand, CIL involves learning to make complex actions, from known actions of human/privileged experts [21, 22, 48]. This issue has been addressed in the recent work 'Learning by Cheating' (LBC) by Chen et al. [15]. LBC utilizes a multi-stage training step, where a privileged (i.e., 'teacher') CIL agent is employed to provide supervision to a non-privileged (i.e., 'student') visuomotor CIL agent. The privileged CIL agent uses a semantic segmentation bird's eye view image as input that is processed by a ResNet and outputs waypoints that are processed by a PID controller to produce the driving controls. The non-privileged agent uses a similar network design but takes as input a frontal camera image, the car's velocity and the conditional command by the navigational planner. As the privileged agent is given access to extensive ground truth information about the world in training and testing, it produces highly plausible and clean trajectories, and thus helps the non-privileged CIL agent to learn how to drive. Learning by Cheating was the first approach to solve the original CARLA benchmark.

In contrast, our framework for learning is very different. In semi-supervised driving with DeepVO, we generate pseduo-labels for diverse out-of-distribution driving scenes and to enable transductive learning of the sensorimotor agent. SelfD, on the other hand, leverages the same visuomotor architecture as teacher and student. We also train in inherently noisy settings, as teacher inference is performed on diverse out-of-distribution image data and not on the original training dataset.

0.2.4 Self-Supervised Visual Representation Learning

0.2.4.1 Self-supervised learning

Self-supervised learning (SSL) is a machine learning approach where the supervisory signal is automatically generated. More precisely, SSL refers to learning data representations by solving a so-called pretext (or auxiliary) task, in a self-supervised fashion, i.e. you automatically generate the supervised signal from the unlabelled data.

0.2.4.2 Contrastive Learning

The core idea of contrastive learning is to attract the positive sample pairs and repulse the negative sample pairs. This methodology has been recently popularized for self-supervised representation learning [60]. Simple and effective instantiations of contrastive learning have been developed using Siamese networks [32, 16, 66]. In practice, contrastive learning methods benefit from a large number of negative samples. These samples can be maintained in a memory bank [60]. In a Siamese network, MoCo [32] maintains a queue of negative samples and turns one branch into a momentum encoder to improve consistency of the queue.

SimCLR [16] directly uses negative samples coexisting in the current batch, and it requires a large batch size to work well.

0.2.4.3 Non-Contrastive Learning

Recent works have shown that we can learn unsupervised features without discriminating between images. Grill et al. [28] propose a metric-learning formulation called BYOL, where features are trained by matching them to representations obtained with a momentum encoder. Methods inspired from BYOL [17, 9], have shown that this method works even without a momentum encoder.

0.2.4.4 Action-COnditioned Policy Pretraining (ACO)

Action-Conditioned policy pretraining paradigm, uses contrastive learning to capture important features in the neural representation relevant to the decision making and benefits downstream tasks. The methods, proposed by Zhang et. al. [70] works by first collecting a large corpus of driving videos with a wide range of weather conditions, from wet to sunny, from all across the world without labeling and then generating action pseudo labels for each frame using a pretrained inverse dynamics model. Then, instead of contrasting images based on different augmented views, this method considers a new contrastive pair conditioned on action similarity and by learning with those action-conditioned contrastive pairs, the representation captures policy-related elements that are highly correlated to the actions. The experimental results show that ACO successfully learns generalizable features for the downstream task such as policy learning through Imitation Learning (IL) and Reinforcement Learning (RL) in end-to-end autonomous driving, and Lane Detection (LD).

In contrast, we focus on non-contrastive learning, particularly, the method DINO proposed by Caron et. al. [9], to learn visual representations from the unlabeled data. These generic representations can then be transferred to the policy learning task.

0.3 Autonomous Driving Framework

We consider the task of point-to-point navigation in an urban setting where the goal is to complete a given route while safely reacting to other dynamic agents and following traffic rules. To achieve this, we consider the imitation learning approach of learning policy, as in self-driving it is easier for an expert to demonstrate the desired behaviour rather than to specify a reward function.

0.3.1 Imitation Learning

The goal of Imitation Learning (IL) is, for an agent to learn a policy $\pi_{\boldsymbol{\theta}}$, that imitates the behavior of an expert π^*. The agent learns to map an input to a navigational decision. In general, the decision may either be a low-level vehicle control action [22] (e.g. steering, throttle and break) or a desired future trajectory relative to the ego-vehicle, i.e., a set of K waypoints [15, 44] in the BEV (birds-eye-view) space. In the latter case, future waypoints may be paired with a hand-specified or learned motion controller to produce the low-level action [15, 44]. In this work, we focus on the later representation due to its interpretability and genralizability. To find the mapping, we consider the Behavior Cloning (BC) approach of IL. To explore different learning approaches in Section 0.4, we would need access to small amount of ground truth data. For that, an expert policy is first rolled out to collect at each time-step, high-dimensional observations of the environment including front camera image, ego-vehicle position and orientation, high-level navigational command and high-level goal location provided as GPS coordinates etc. From these high-dimensional observations, we derive our dataset $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{W}_i)\}_{i=1}^{N} \in (\mathcal{X}, \mathcal{W})$ of size N, where the input $\mathbf{X}$ consists of the front camera image and the goal location and the corresponding expert trajectory $\mathbf{W}$, defined by a set of 2D waypoints relative to the coordinate frame of the ego-vehicle in BEV space, i.e., $\mathbf{W} = \{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^{T}$, are calculated from the ego-vehicle positions and orientations from the subsequent frames. Our goal is to find a decision making policy i.e. a waypoint prediction function $\pi_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{W}$ with learnable parameters $\boldsymbol{\theta} \in \mathbb{R}^d$. In BC, the policy $\pi_{\boldsymbol{\theta}}$ is learned by training a neural network in a supervised manner using the dataset, $\mathcal{D}$, with a loss function, $\mathcal{L}$ i.e.

$$\underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{(\mathbf{X},\mathbf{W})\sim\mathcal{D}}[\mathcal{L}(\mathbf{W}, \pi_\theta(\mathbf{X}))].$$

We use the L_1 distance between the predicted trajectory, $\pi_{\boldsymbol{\theta}}(\mathbf{X})$, and the corresponding expert trajectory, $\mathbf{W}$, as the loss function. We assume access to an inverse dynamic model [5], implemented as a PID controller $\mathbb{I}$, which performs the low-level control, i.e., steer, throttle and brake, provided the future trajectory $\mathbf{W}$. The action are dertermined as $\mathbf{A} = \mathbb{I}(\mathbf{W})$.

0.3.2 Input and Output Representations

0.3.2.1 Input Representation

0.3.2.1.1 Driving Scenes

We note that, even though our collected expert demostrations from CARLA and the YouTube driving videos are sequential, we do not use temporal data for training. Contrary to our intuition of getting better generalization in decision-making from sequential observations, the prior works on IL for autonomous driving have shown that using observation histories may not lead to performance gain [30, 43, 3, 56]. Thus, we use a single time-step input. We consider the front camera with a FOV of $120°$. We extract the front image at a resolution of 960×480

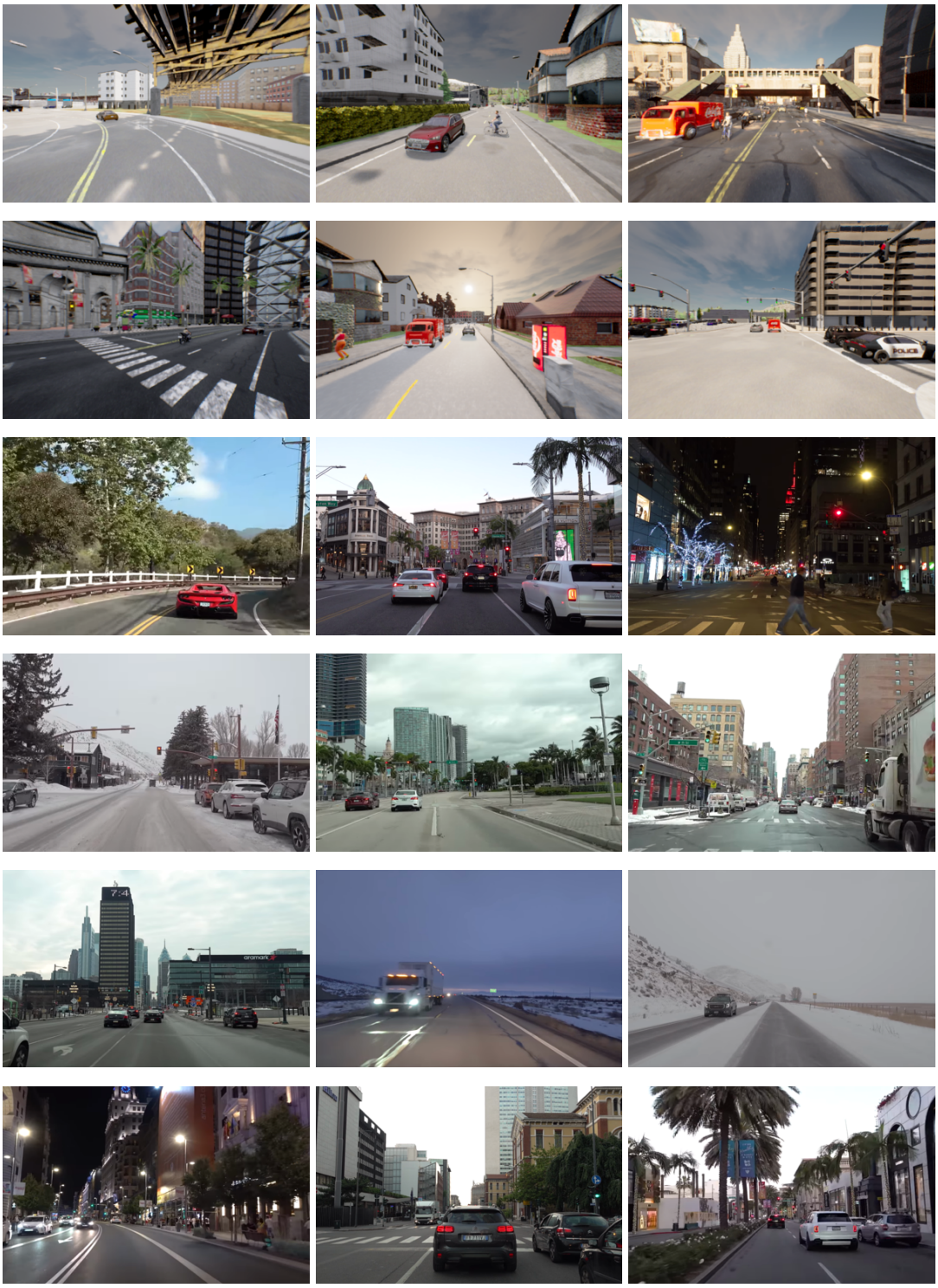

Figure 1: First two rows shows examples of driving scenes from the CARLA simulator and rest of the images are extracted from the YouTube driving videos.

pixels which we resize and crop to 256×256 to remove radial distortion at the edges. Figure 1 shows some example driving scenes from CARLA and the YouTube driving videos.

0.3.2.1.2 Global Planner

We follow the standard protocol of CARLA 0.9.10 and assume that high-level goal locations c are provided as GPS coordinates by an A^* navigational planner. Agents are supposed to follow routes directed by these GPS coordinates. Note that, these goal locations are sparse and can be hundreds of meters apart, as opposed to the local waypoints predicted by the policy π_{θ}.

0.3.2.2 Output Representation

We predict the future trajectory $\mathbf{W}$ of the ego-vehicle in BEV space, centered at the current coordinate frame of the ego-vehicle. The trajectory is represented by a sequence of 2D waypoints, $\mathbf{W} = \{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^{T}$. We use $T = 4$, which is the default number of waypoints required by our inverse dynamics model.

0.3.3 Waypoint Prediction Network

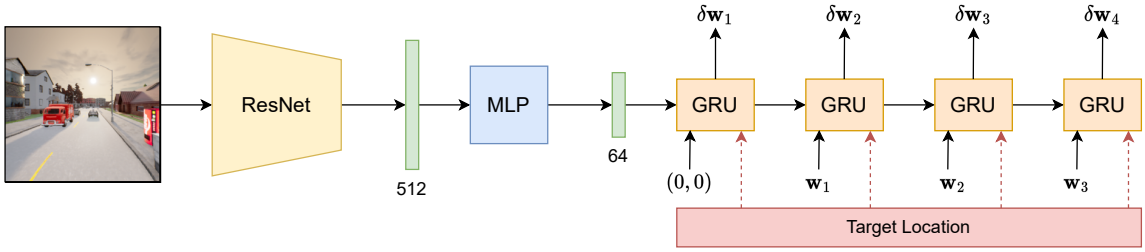

Figure 2: Architechture of the waypoint prediction netowork AIM [48]

We adpot the image-based basline network architechture 'AIM' proposed by Prakash et. al. [48]. As shown in Figure 2, first, we encode the input driving scene with a ResNet34 architechture into a 512-dimensional feature-encoding. We then pass this 512-dimensional feature vector through an MLP (comprising 2 hidden layers with 256 and 128 units) to reduce its dimensionality to 64 for computational efficiency before passing it to the auto-regressive waypoint network implemented using GRUs [19]. We initialize the hidden state of the GRU with the 64-dimensional feature vector. The update gate of the GRU controls the flow of information encoded in the hidden state to the output and the next time-step. It also takes in the current position and the goal location (Section 0.3.1) as input, which allows the network to focus on the relevant context in the hidden state for predicting the next waypoint. We provide the GPS coordinates of the goal location (transformed to the ego-vehicle coordinate frame) as input to the GRU rather than the encoder since it lies in the same BEV space as the predicted

waypoints and correlates better with them compared to representing the goal location in the perspective image domain [15]. Following [24], we use a single layer GRU followed by a linear layer which takes in the hidden state and predicts the differential ego-vehicle waypoints $\{\delta\mathbf{w}_t\}_{t=1}^{T}$ for $T = 4$ future time-steps in the ego-vehicle current coordinate frame. Therefore, the predicted future waypoints are given by $\{\mathbf{w}_t = \mathbf{w}_{t-1} + \delta\mathbf{w}_t\}_{t=1}^{T}$. The input to the first GRU unit is given as $(0, 0)$ since the BEV space is centered at the ego-vehicle's position.

0.3.4 PID Controller

We use two PID controllers for lateral and longitudinal control to obtain steer, throttle and brake values from the predicted waypoints, $\{\mathbf{w}_t\}_{t=1}^{T}$. The longitudinal controller takes in the magnitude of a weighted average of the vectors between waypoints of consecutive time-steps whereas the lateral controller takes in their orientation. For the PID controllers, we use the same configuration as in the author-provided codebase of [15].

0.3.5 Loss Function

We train the network using an L_1 loss between the predicted waypoints and the ground truth waypoints (from the expert), registered to the current coordinate frame. Let $\mathbf{w}_t^{\text{gt}}$ represent the ground truth waypoint for time-step t, then the loss function is given by:

$$\mathcal{L} = \sum_{t=1}^{T} \left\| \mathbf{w}_t - \mathbf{w}_t^{\text{gt}} \right\|_1 .$$

Note that, the ground truth waypoints $\left\{\mathbf{w}_t^{\text{gt}}\right\}$ which are available only at training time are different from the sparse goal locations c provided at both training and test time.

0.4 Learning Approaches

Our goal is to facilitate training driving policies at scale. We want to efficiently make use of the broad and diverse experience found in large amounts of unlabeled videos. To do so, in this section we explore a few semi-supervised and self-supervised learning approaches. First, we introduce our proposed semi-supervised learning method based on deep visual odometry. In the next section, we discuss an existing semi-supervised learning approach "SelfD" proposed Zhang et. al. [69] and incorporate it in our autonomous driving framework. Lastly, we incorporate and explore in our self-driving framework, "OVRL", a self-supervised learning approach proposed by Yadav et. al. [63] which was studied in the domain of embodied navigation.

0.4.1 SemiD: Semi-supervised Driving with Deep Visual Odometry

In this section, we will introduce our proposed method for semi-supervised learning for autonomous driving. Our key idea is to generate pseudo-labels for unlabeled driving scenes by exploiting deep visual odometry, a learning-based ego-motion estimation method, because of the following reasons:

- It is robust to image noise and independent of camera calibration.
- In our expert driving dataset, the weather conditions and the time of the day changes randomly from frame to frame. For this reason learning-based method is a better choice for ego-motion estimation than the classical methods.
- In our driving framework, we use a set of waypoints for low-level control and a target point as a high-level navigational command. Intuitively, we can understand that, for good driving performance, one requires accurately predicted waypoints and a rough estimate of the target location. Our collected pseduo-labels satisfies this criteria as the estimated waypoints with deep visual odometry are fairly accurate due less error accumulation in the beginning stages of estimation and the rough estimation of the target point is sufficient as a high-level navigational command.
- Visual odometry only estimate the relative motion between two frames to recover the expert's action from the image sequence and therefore we do not have to rely on the model's understanding of the driving.

0.4.1.1 Deep Visual Odometry

We use an end-to-end learning approach following [58, 67] to train the model to map directly from input image pairs to an estimate of ego-motion (in our use case, estimate relative translation is sufficient). The model we used, is a two module Long-term Recurrent Convolutional Neural Networks. The feature-encoding module encodes the short-term motion feature in a pair of images, while the memory-propagating module captures the long-term motion feature in the consecutive image pairs.

0.4.1.1.1 Feature-encoding Module

In order to learn the geometric relationships from two adjacent images, we use the following CNN architechture, inspired by FlowNetSimple architechture [25] ignoring the decoder part of it and only focusing the on the convolutional encoder. Contrary to DeepVO [58] and PoseConvGRU [67], which uses huge 10-layered CNN architechtures, we use a very simple and lightweight 5-layered CNN architechture as shown in Table 1. Each layer is followed by an application of ReLU nonlinear activation function. We keep the kernel size to 3, padding size to 1 for all the layers. The channel dimension doubles in each subsequent layers. We use maxpool of size 2 in the first 2 layers and use maxpool of size 4 for the rest of the layers.

Layer	Kernel Size	Padding	Stride	Max Pool	Number of Channels
Input	-	-	-	-	6
Conv1	3×3	1	0	2×2	64
Conv2	3×3	1	0	2×2	128
Conv3	3×3	1	0	4×4	256
Conv4	3×3	1	0	4×4	512
Conv5	3×3	1	0	4×4	1024

Table 1: Configuration of the CNN

The reason behind this is that, having small receptive field in the first 2 layers encourages the network to learn about the fine-grained geometric details in the pair of images which is essential for relative motion estimation. On the other hand, the large receptive field in the later layers enforces the network to ignore the global context and also helps in reducing the number of learnable parameters as well. The input to the CNN are a sequence of $n + 1$, 256×256 RGB driving scenes from CARLA. With $n + 1$ sequential driving scenes, we can obtain n sets of image pairs taking two adjacent frames at a time. These image pairs are then fed to the 5-layered CNN to obtain a feature maps of size $1 \times 1 \times 1024$ for each image pair. Contrary to the typical training process with augmented data for CNNs, we only use the original images for accurate relative motion estimation, because performing any pre-processing operation to the images such as blurring, adding noise, random clipping etc., can hamper the network to learn the geometric relationship of the objects in the images.

0.4.1.1.2 Memory-propagating Module

Following [67], we use a stacked GRU (Gated Recurrent Unit) [2] as our memory-propagating module as showns in Figure 3. The memory module builds a set of chronological visual representations from the CNN embeddedings of the sequence of image pairs. Because of its ability of remember histories, GRU can capture the geometric relationships coming from the previous frames of images, and then estimate the relative motion for the current frame utilizing the geometric constraint within multiple frames. Also, GRUs are appropriate for this module as they are simple yet powerful. They contain fewer gates compared to LSTMs (Long Short-Term Memory unit) and thus reduce the number of learnable parameters and, yet, provide similar performance as LSTMs [20]. In our implementation, we flatten the $1 \times 1 \times 1024$-dimensional CNN embeddeding into a feature vector, which we then further process through a fully connected layer and reduce its dimensions to 256 for computational efficiency before passing it to the GRU. Following [24], we use a single layer GRU followed by a linear layer which takes in the hidden state and predicts the relative translation of the ego-vehicle implied

by the pair of images. Finally, we accumulate all the relative translations to compute the ego-vehicle trajectory.

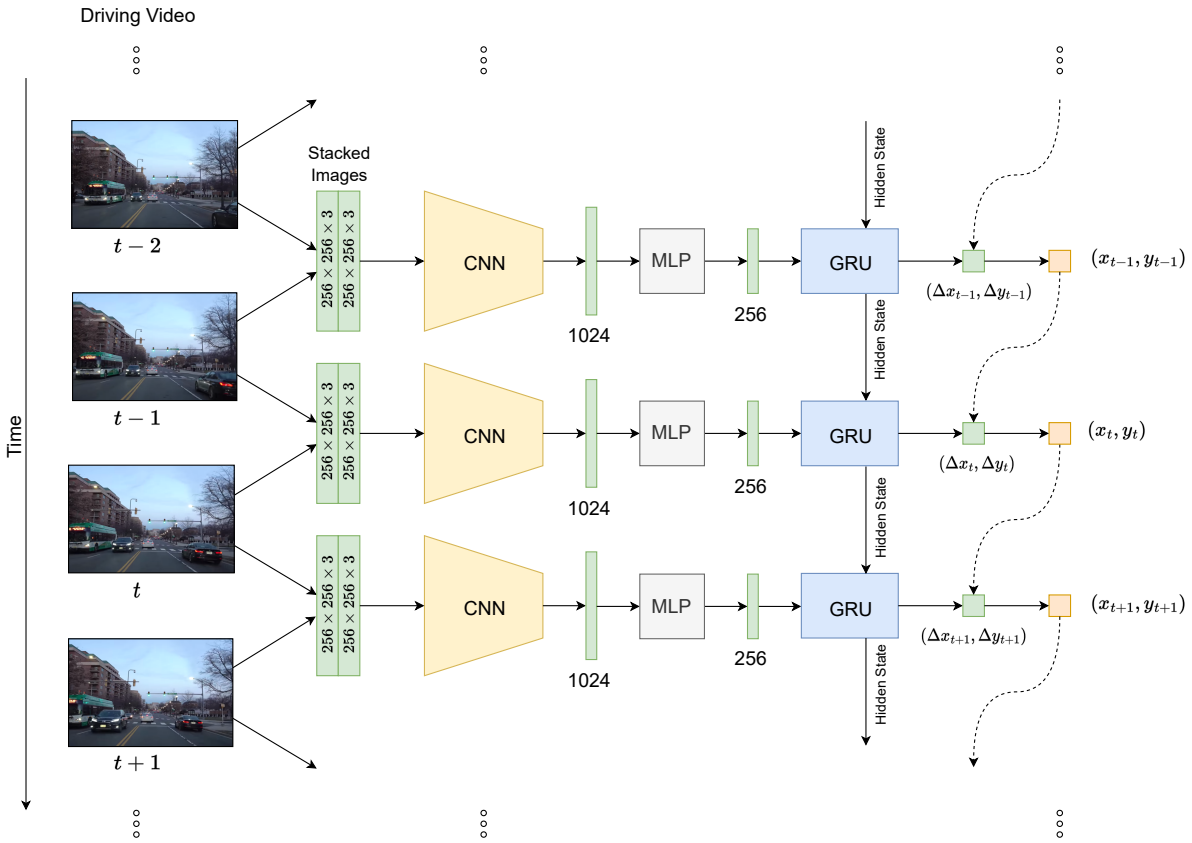

Figure 3: Architecture of the ConvGRU-based monocular VO system.

0.4.1.2 Generating Pseduo-labels for Video frames

The training of the deep visual odometry network assumes access to a small labeled dataset $\mathcal{D}$ (as discussed in Section 0.3.1), which includes driving scenes and its corresponding ego-vehicle absolute postions, to learn the relative motion estimation. This assumption is reasonable considering that there are several publicly available driving datasets containing ego-vehicle position and orientation for relative motion estimation [26, 59]. At the inference time, we compute the ego-vehicle trajectory with the following recursive formulation.

Let, the position and orientation of the ego-vehicle at the 0-th time-step is at the origin of the coordinate system with its heading in the direction of the x-axis i.e., $x_0 = 0, y_0 = 0, \cos\theta_0 = 1, \sin\theta_0 = 0$. Then, we can compute the ego-vehicle position and orientation at the $t+1$-th timestep given its position and orientation at t-th timestep as follows:

- $\Delta x_{t+1}, \Delta y_{t+1}$ be the predicted relative motion from timestep t to $t+1$

- $\cos(\Delta\theta_{t+1}) = \frac{\Delta x_{t+1}}{\sqrt{\Delta x_{t+1}^2 + \Delta y_{t+1}^2}}$
- $\sin(\Delta\theta_{t+1}) = \frac{\Delta y_{t+1}}{\sqrt{\Delta x_{t+1}^2 + \Delta y_{t+1}^2}}$
- $\cos(\theta_{t+1}) = \cos(\theta_t + \Delta\theta_{t+1}) = \cos(\theta_t)\cos(\Delta\theta_{t+1}) - \sin(\theta_t)\sin(\Delta\theta_{t+1})$
- $\sin(\theta_{t+1}) = \sin(\theta_t + \Delta\theta_{t+1}) = \sin(\theta_t)\cos(\Delta\theta_{t+1}) + \cos(\theta_t)\sin(\Delta\theta_{t+1})$
- $x_{t+1} = x_t + \cos(\theta_{t+1})\sqrt{\Delta x_{t+1}^2 + \Delta y_{t+1}^2}$
- $y_{t+1} = y_t + \sin(\theta_{t+1})\sqrt{\Delta x_{t+1}^2 + \Delta y_{t+1}^2}$

Now, given a sequence of driving scenes (e.g. 30), we use the above recursive formulation and get the corresponding ego-vehicle trajectory. Now, due error accumulation, it is evident that as we progressively aggregate more and more relative ego-motions over a longer time horizon, the estimated trajectory gradually drifts from the ground truth trajectory. But for our use case, we only need accurate trajectory estimation for the first 4 steps (as we use 4 waypoints for low-level vehicle control) which holds true due to less error accumulation in the initial few steps. Also, a rough understanding of the target location is sufficient for driving and so, we take the end point of the trajectory as an estimate of the target point. Thus, we generate pseduo-labels i.e. pseudo-waypoints and pseduo-target-point for the initial driving scene. We repeat this procedure for all the unlabeled driving scenes and thus we create our pseduo-labeled dataset $\hat{\mathcal{D}}_{\text{SemiD}}$. Figure 4 illustrates some example driving scenes from CARLA and YouTube videos and their corresponding pseduo-waypoints estimated by deep visual odometry.

0.4.1.3 Model Pre-Training and Fine-Tuning

We pretrain the waypoint network $\pi_{\boldsymbol{\theta}}$ from scratch over the large and diverse pseduo-labeled dataset $\hat{\mathcal{D}}_{\text{SemiD}}$. The pre-trained policy can then be further fine-tuned over the small original dataset $\mathcal{D}$. Thus, we leverage these noisy but acceptable informations from the pseduo-labeled dataset $\hat{\mathcal{D}}_{\text{SemiD}}$ to help the network gain a overall knowledge about the mapping and then we provide the network with the accurate informations through our small original dataset $\mathcal{D}$ to enable the network to learn the mapping from the ground-truth information.

0.4.2 SelfD: Self-Learning Large-Scale Driving Policies

In this section, we discuss the semi-supervised approach "SelfD" proposed by Zhang et al. [69]. We follow the three main steps. We use a monocular image-based waypoint network AIM (Section 0.3.3) that reasons directly in the BEV. Next, we look into the proposed data augmentation step for obtaining multiple plausible pseudo-labels when self-training over unlabeled YouTube data (Section 0.4.2.5). Finally, we re-train the model over the larger dataset (Section 0.4.2.6).

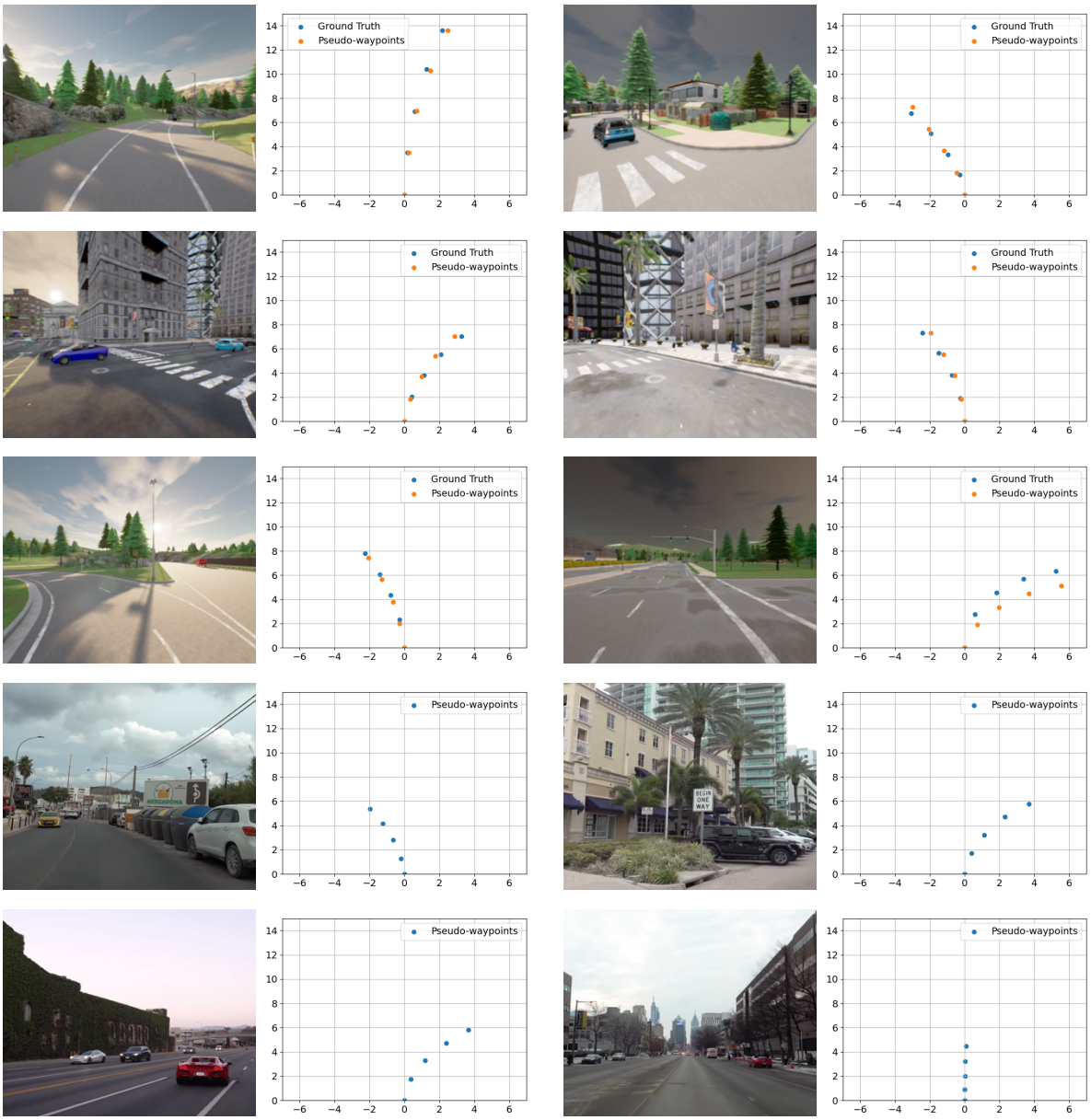

Figure 4: The first three rows shows examples of driving scenes from the CARLA simulator and its corresponding ground-truth waypoints and the pseduo-waypoints estimated by the DeepVO method. The later two rows shows the estimated pseduo-waypoints of some images extracted from the YouTube driving videos.

0.4.2.1 Conditional Imitation Learning from Observations

To make use of the unlabeled driving scenes containing diverse navigational experience, the authors propose 'Conditional Imitation Learning from Observations' (CILfO) [69] framework. The framework suggests a way to generate pseudo-labels for the unlabeled driving scenes. But, the network architechture used by the authors is similar to that of LBC [15] (uses discrete navigational command as the conditional command). In the work of Prakash et. al. [48], the authors have shown that, their proposed image-based architechture, AIM (uses sparse goal location as the conditional command), performs significantly better than the LBC architechture. Therefore, in this work, we explore the CILfO framework in our autonomous driving setting which uses AIM in its core (Section 0.3.3). Therefore, we consider sparse goal location as the conditional command instead of the discrete navigational commands. Therefore, utilizing the CILfO framework, we can then recover the waypoints $\hat{\mathbf{W}}$, and the goal location $\hat{c}$ from the input image to construct a dataset

$$\hat{\mathcal{D}}_{\text{SelfD}} = \left\{ \left((\mathbf{I}_i, \hat{c}_i), \hat{\mathbf{W}}_i \right) \right\}_{i=1}^{M}$$

which can be then used to train a policy using behaviour cloning.

0.4.2.2 Initial Data Assumption

The CILfO learning task assumes access to a small labeled dataset to learn an initial policy mapping using human expert demonstrations. We then use this learned policy to gather pseduo-labels for the unlabeled data. This assumption is reasonable considering that there are several publicly available driving datasets with included action labels [26, 59]. In this work, we use the small subset of a labeled dataset $\mathcal{D}$ collected by running an expert policy roll out in CARLA [23] as discussed in Section 0.3.1.

0.4.2.3 Self-supervised Training Process

In this section, we discuss the proposed generalized training method for leveraging unconstrained and unlabeled demonstration data. The proposed semi-supervised policy training process, SelfD, can be learned in three summarized steps:

1. Use a small, labeled domain-specific dataset $\mathcal{D}$ to learn an initial observations-to-BEV policy $\pi_{\boldsymbol{\theta}}$ via imitation learning.
2. Obtain a large pseudo-labeled dataset $\hat{\mathcal{D}}_{\text{SelfD}}$ by leveraging sampling from $\pi_{\boldsymbol{\theta}}$.
3. Pre-train a generalized policy $\pi_{\boldsymbol{\theta}}$ on $\hat{\mathcal{D}}_{\text{SelfD}}$ and fine-tune on the clean labels of $\mathcal{D}$.

0.4.2.4 BEV Plan Network

To account for arbitrary cameras, viewpoints and scene layouts, the authors proposed a monocular planner that predicts the waypoints in the image plane and then uses a MLP to transform

those waypoints in the BEV-space. In our work, instead of proposed method, we use an equivalent alternative. We use a GRU to predict a future plan parameterized by waypoints, directly in the BEV plan space as discussed in Section 0.3.3. Besides using two output nodes for predicting the waypoints, we also add one extra output node to train the augmented network to predict the quality estimates $\sigma \in [0, 1]$ of the waypoints. We use the IOU of the bounding boxes of the ground-truth and the predicted waypoints as a proxy for the quality estimates of the waypoints. Therefore, our training loss function is defined as

$$\mathcal{L} = \mathcal{L}_{\text{waypoints}} + \mathcal{L}_{\text{quality}}$$

where $\mathcal{L}_{\text{waypoints}}$ is the L_1 distance between the ground-truth and the predicted waypoints as discussed in 0.3.5 and $\mathcal{L}_{\text{quality}}$ is a binary cross-entropy loss.

0.4.2.5 "What If" Pseudo-Labeling of the Unlabeled Data

Given a set of unlabeled images $\mathcal{U}$, we sample from the trained conditional policy $\pi_{\boldsymbol{\theta}}$ in a semi-supervised training process. Contrary to our proposed method in Section 0.4.1, the authors do not consider visual odometry techniques [58] to recover the expert's actions, as they found these result in highly noisy trajectories in their online video settings. Thus, they propose to leverage a single-frame pseudo-labeling mechanism i.e. to employ the conditional model $\pi_{\boldsymbol{\theta}}$ to generate multiple hypothetical future trajectories in a process referred to as "what if" augmentation. Beyond resolving the missing target point, the proposed augmentation claims to provides additional supervision, i.e., a conditional agent that better reasons on what it might need to do, for instance, if it had to turn left instead of right at an intersection (Figure 5). The procedure to gather "what if" augmentation is as follows:

- sample the target point $\hat{c}$ uniformly from a half disk of radius 50 meters.
- use the conditional model to gather pseudo-labels $(\hat{\mathbf{W}}, \hat{\sigma}) = \pi_{\boldsymbol{\theta}}(I, \hat{c})$.
- discard noisy trajectories with some thresholding on the quality estimate $\hat{\sigma}$ and take the rest to form the pseduo-labeled dataset $\hat{\mathcal{D}}_{\text{SelfD}}$.

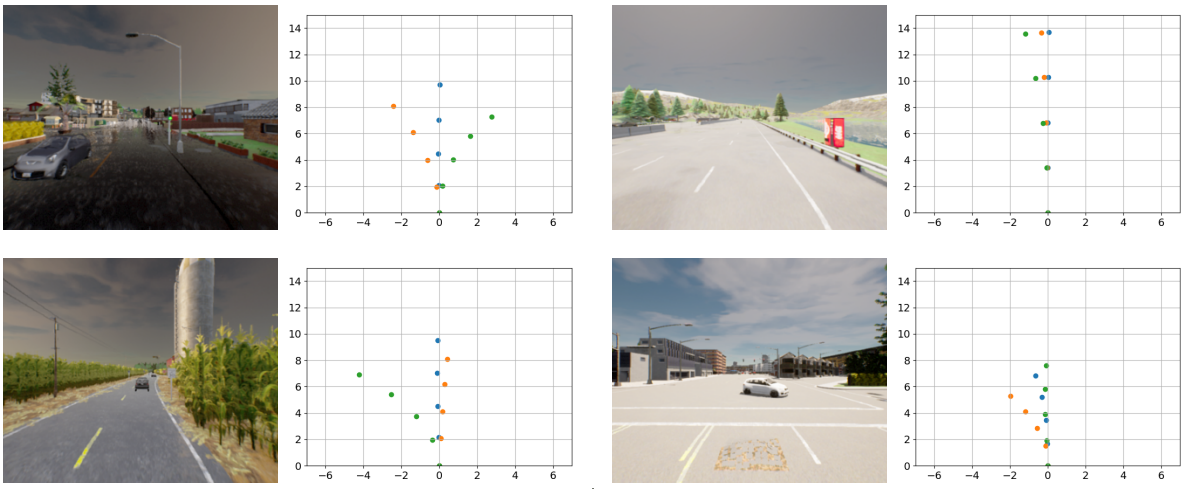

Figure 5: **"What If" Hypothetical Pseudo-Labeling.** We generate multiple plausible future trajectories in BEV for each unlabeled frame by unformly sampling the target point location. Here we illustrate what if pseduo-labeling for two driving scenarios from our CARLA dataset.

0.4.2.6 Model Pre-Training and Fine-Tuning

As a final training step, we re-train the waypoint network $\pi_{\boldsymbol{\theta}}$ from scratch over the large and diverse dataset $\hat{\mathcal{D}}_{\text{SelfD}}$. The pre-trained policy can then be further fine-tuned over the original dataset $\mathcal{D}$.

0.4.3 OVRL: Offline Visual Representation Learning

In this section, we discuss OVRL[63], a two-stage learning approach proposed by Yadav et. al. and incorporate it in our autonomous driving framework. As an overview, this learning approach includes an encoder pretraining step using DINO, followed by downstream policy learning via behavior cloning.

0.4.3.1 Self-supervised Pretraining

As the first step, a visual encoder is pretrained using DINO [9], a simple but effective self-supervised learning algorithm. DINO uses knowledge distillation as a mechanism for self-training, where the student network $g_{\boldsymbol{\theta}_s}$ is trained to match the output of the teacher network $g_{\boldsymbol{\theta}_t}$. As illustrated in Figure 6, it takes an input image x and using the multi-crop strategy introduced in [10], it generates multiple distorted views or crops from it, specifically, it produces two global views (x_1^g and x_2^g) at 224×224 resolution and eight local views x^l at a lower resolution (96×96). All crops are passed through the student network while only the global views are passed through the teacher network. The student and teacher networks both output K dimensional feature vectors for each view, which are converted into probability

distributions (P_s and P_t) using a temperature scaled softmax function as follows:

$$P_s(x)^{(i)} = \frac{\exp\left(g_{\boldsymbol{\theta}_s}(x)^{(i)}/\tau_s\right)}{\sum_{k=1}^{K}\exp\left(g_{\boldsymbol{\theta}_s}(x)^{(k)}/\tau_s\right)}$$

where the temperature parameter τ_s controls the sharpness of the output distribution, and a similar formula holds for P_t with temperature τ_t. Now, given a fixed teacher network $g_{\boldsymbol{\theta}_t}$, the student network learns to match the distribution of the teacher network by minimizing the cross-entropy loss:

$$\mathcal{L}(\boldsymbol{\theta}_s) = \sum_{x\in x_1^g, x_2^g} \sum_{x'\in\{x_1^g, x_2^g\}\cup\{x_i^l\}_{i=1}^{8}} P_t(x)\log(P_s(x')).$$

The teacher network parameters are updated as an exponential moving average of the student network parameters.

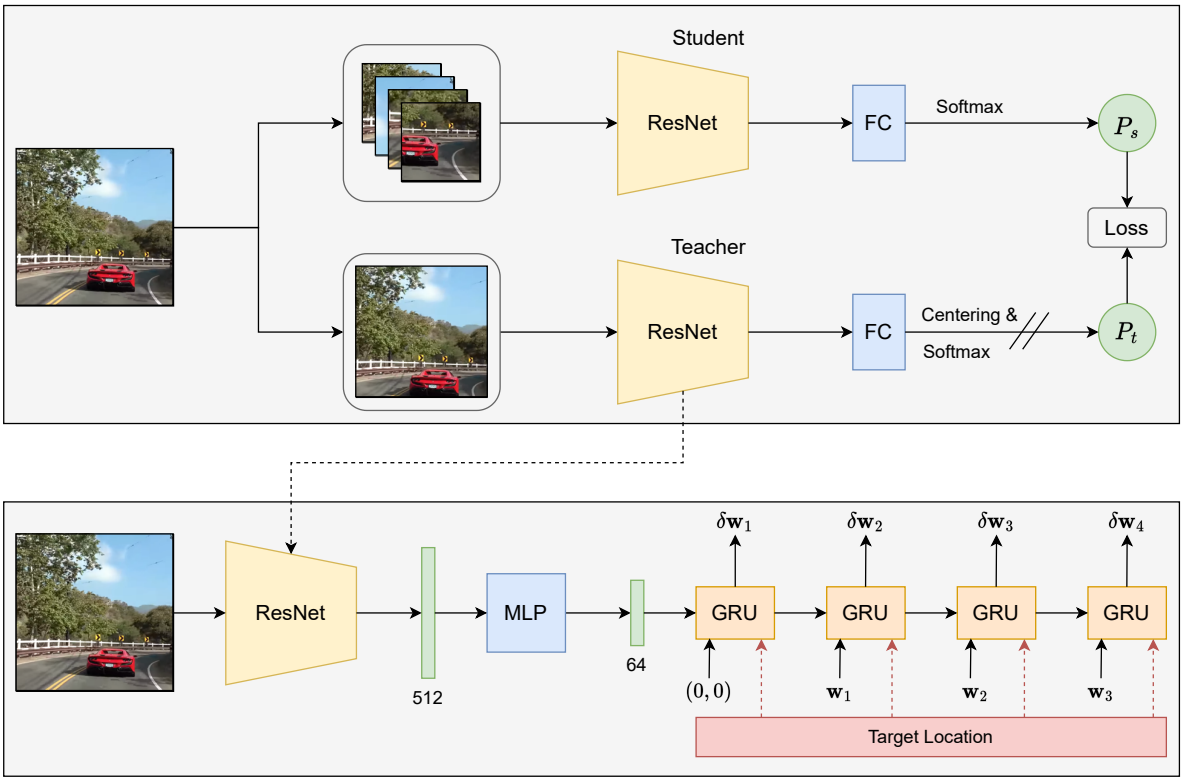

Figure 6: Overview of OVRL, consisting of two steps: 1) offline pretraining of the visual representations using large-scale pre-rendered images of indoor environments using DINO 2) downstream finetuning of the visuomotor representations on the ImageNav RL task in Habitat.

Representation learning by self-supervised learning algorithm is always prone to collapse, i.e., the network can converge to a trivial solution by predicting the same representation for

every image. To avoid collapse, DINO centers and sharpens the teacher's output before the softmax operation. Specifically, the centering operation adds the term c to the teacher's output, which is updated as follows: $c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=0}^{B} g_{\boldsymbol{\theta}_t}(x_i)$, where $m > 0$ is a momentum parameter and B is the batch size. Sharpening is achieved by setting low value for the temperature parameter τ_t for the teacher softmax normalization. Thus, centering prevents one dimension to dominate but encourages collapse to the uniform distribution, while the sharpening has the opposite effect. Applying both operations balances their effects which is sufficient to avoid collapse in presence of a momentum teacher.

0.4.3.2 Implementation Details

We use the same ResNet34 architechture as discussed in 0.3.3. We pretrain the model on the CARLA driving scenes without the labels. We train with the AdamW optimizer [40] and a batch size of 16 per-gpu, distributed over 8 GPUs. The learning rate is linearly ramped up during the first 10 epochs to its base value determined with the following linear scaling rule [27]: lr = 0.0005 $\times$ batchsize/256. After this warmup, we decay the learning rate with a cosine schedule [41]. The weight decay also follows a cosine schedule from 0.04 to 0.4. The temperature τ_s is set to 0.1 while we use a linear warm-up for τ_t from 0.04 to 0.07 during the first 30 epochs. We follow the data augmentations of BYOL [28] (color jittering, Gaussian blur and solarization) and the multi-crop strategy [10].

0.4.3.3 Downstream Learning

After pre-training with DINO, the whole projection head is discarded. Now we plug this pretrained ResNet34 architechture in our waypoint prediction network (Section 0.3.3). Now this waypoint prediction network is further fine-tuned over the small original dataset $\mathcal{D}$. Thus, we leverage unlabeled driving scenes to help the visual encoder attain a better representation of the input and then we provide the network with the navigational informations through our small original dataset $\mathcal{D}$ and thus enables the network to quickly map the input scenes to the waypoints leveraging the better internal representation.

0.5 Experimental Results

In this section, we provide our experimental setup, compare the driving performance of our approach against the prior works, present an ablation study exploring different unlabeled datasets and learning approaches and conduct an infraction analysis to study different failure cases.

0.5.1 Task

We consider the task of navigation along a set of predefined routes in a variety of areas, e.g. freeways, urban areas, and residential districts. The routes are defined by a sequence of sparse goal locations in GPS coordinates provided by a global planner and the corresponding discrete navigational commands, e.g. follow lane, turn left/right, change lane. Our approach uses only the sparse GPS locations to drive. Each route consists of several scenarios, initialized at predefined positions, which test the ability of the agent to handle different kinds of adversarial situations, e.g. obstacle avoidance, unprotected turns at intersections, vehicles running red lights, and pedestrians emerging from occluded regions to cross the road at random locations. The agent needs to complete the route within a specified time limit while following traffic regulations and coping with high densities of dynamic agents. Infractions that are penalized are:

- Collision with a pedestrian
- Collision with a vehicle
- Collision with a static object
- Running a red light
- Running a stop sign
- Driving on the wrong lane or sidewalk
- Leaving the route specified by the navigational planner

Along the routes, there are dangerous scenarios specified which the agent needs to resolve in order to safely arrive at his destination. There are currently 10 different scenarios specified:

- Rubble on the road leading to a loss of control.
- Leading vehicle suddenly performs an emergency brake.
- A pedestrian hidden behind a static object suddenly runs across the street.
- After performing a turn at an intersection, a cyclist suddenly drives across the street.
- A slow vehicle gets spawned in front of the agent.
- A static object is blocking the street.
- Crossing traffic is running a red light at an intersection
- The agent must perform an unprotected left turn at an intersection with oncoming traffic
- The agent must perform a right turn at an intersection with crossing traffic

- Crossing an unsignalized intersection.

0.5.2 Implementation Details

0.5.2.1 Data Cleansing

Our expert driving demostrations contain many failure cases. For examples, ego-vehicle crashes into another vehicle or another vehicle comes infront of the ego-vehicle while its taking a turn. For most of these failure cases, the ego-vehicle does not move after the incident for the rest of frames in that route demonstration. Also, in the beginning of the route demonstrations, the ego-vehicle takes some time to reach the desired speed from zero, even though from the BC perspective the ego-vehicle should be on full speed in such situation as there is no car infront of it. These failure cases introduce spurious corrections that lead to causal confusion problem [22]. Therefore we remove these demonstrations which leads to improved driving performance.

0.5.2.2 Stratified Sampling

On top of cleaning the data, we also group the data according to different driving modalities. For example, dependending on the waypoints corresponding to a driving frame, we catogorize the sample as one of the four categories — 'at turns', 'straight driving', 'at halt' or 'accelerating'. Next, we implement stratified sampling technique in our dataloader, which not only use the good samples but also generate a training batch (we use batch size of 64 in all our experiments) consisting of equal number of examples from each modalities. We do this because in our CARLA dataset and specifically in the YouTube dataset, we observe that not all modalities occur equally, for example number of instances where the car turns are very rare. This sampling technique mitigates this issue and therefore improves the driving perfomance.

0.5.2.3 Image Augmentation

Prior works [35, 34, 65, 42] have shown that using image augmentations during policy learning can help improve overall performance and leads to better generalization on the test set. Even though, we have not exhaustively searched for the optimal augmentations, but it is intuitively understandable that heavy image augmentation will degrade driving performance (for example, adding heavy Gaussian noise can interfere with the traffic light signal). Thus, we use mild augmentations such that they affect images very little. We use Gaussian noise, Gaussian blur, Dropout, Salt and Paper noise, which lead to improved robustness of the model.

0.5.3 Dataset

We use the CARLA [23] simulator for training and testing, specifically CARLA 0.9.10 which consists of 8 publicly available towns. We use 7 towns for training and hold out Town05 for

evaluation. For generating training data, we roll out an expert policy designed to drive using privileged information from the simulation and store data at 2 FPS. We select Town05 for evaluation due to the large diversity in drivable regions compared to other CARLA towns, e.g. multi-lane and single-lane roads, highways and exits, bridges and underpasses. We consider Town05-Long evaluation setting: 10 long routes of 1000-2000m comprising 10 intersections each. Each route consists of a high density of dynamic agents and adversarial scenarios which are spawned at predefined positions along the route. Since we focus on handling dynamic agents and adversarial scenarios, we decouple this aspect from generalization across weather conditions and evaluate only on ClearNoon weather.

We also use a set of first-view driving videos from YouTube collected by the Zhang et. al. [70]. We use 52 videos with a total length of over 50 hours of driving demonstration. As shown in Figure 1, these videos cover different driving scenes with various weather conditions (sunny, rainy, snowy, etc.) and regions (rural and urban areas). We sample two frames every one second, resulting to a dataset of 0.36 million frames. We use all the the YouTube driving data for pretraining the models in all learning approaches.

0.5.4 Evaluation Metrics

For the CARLA Autonomous Driving Leaderboard, the driving performance of an agent is characterized by a set of chosen metrics that considers different aspects of driving. While all routes have the same type of metrics, their respective values are calculated separately. These metrics are as follows:

0.5.4.1 Route Completion (RC)

Route Completion is the percentage of the route that is completed by an agent. If an agent drives off-road, that percentage of the route will not be considered towards the computation of the route completion score. Additionally, the following events will interrupt the simulation, preventing the agent to continue which will effectively reduce the route completion:

- Route deviation: If an agent deviates more than 30 meters from the assigned route.
- Agent blocked: If an agent doesn't take any actions for 180 simulation seconds.
- Simulation timeout: If no client-server communication can be established in 60 seconds.
- Route timeout: If the simulation of a route takes too long to finish.

0.5.4.2 Infraction Score (IS)

Infraction Score is a penalty for infractions where the agent starts with an ideal base score of 1.0. For every infraction, the score is multiplied by the penalty coefficient of that infraction

type. Ordered by their severity, the penalty coefficients are as follows:

- Collision with a pedestrian: 0.50
- Collision with a vehicle: 0.60
- Collision with a static object: 0.65
- Running a red light: 0.70
- Running a stop sign: 0.80

Note that this means that subsequent infractions will have a lower impact due to the multiplicative nature of the score.

0.5.4.3 Driving Score (DS)

Driving Score is the main metric for performance, serving as the product between the route completion and the infractions penalty. It is calculated in the following way:

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

where N is the number of routes, R_i the route completion percentage of the i-th route and P_i the infraction penalty of i-th route. Note that, this is not the same as multiplying the averaged route completion with the averaged infraction score. The driving score is a normalized metric, meaning that the best possible score is 100 and the worst score is 0. For the validation routes, we run them with 3 different seeds and report the mean and standard deviation of the 3 scores averaged across the routes.

0.5.5 Results

0.5.5.1 Performance of Supervised Training

In our first experiment, we examine to what extent we can improve the current image-based AIM architechture [48] in CARLA evaluation setting involving complex multi-lane intersections, adversarial scenarios, and heavy infraction penalties. In Table 2, we report the evaluation results of supervised training with all the 7 CARLA training towns (Section 0.5.3) over 3 random seeds. We observe that smaller learning rate improve the driving performance. Next, we train the model with the filtered data and train with stratified sampling enabled dataloader. Here again we get an improvement in driving performance by a small margin. Lastly, we introduce image augmentation in our training. We find that image augmentation helps improve the performance by a large margin. We note that each of these techniques improve the driving independently. Thus, combining all these techniques, we gain significant improve in the driving perfomance cumulating their individual contributions.

ImgAug	Cleaned Data + Stratified Sampling	lr	DS (↑)	RC (↑)	IS (↑)
✗	✗	1e-4	13.39 ± 1.81	60.31 ± 6.11	0.35 ± 0.02
✗	✗	2e-5	17.57 ± 3.34	68.75 ± 5.70	0.32 ± 0.04
✗	✓	2e-5	20.91 ± 2.83	71.26 ± 4.84	0.34 ± 0.01
✓	✗	2e-5	28.55 ± 3.92	85.47 ± 9.23	0.36 ± 0.07
✓	✓	2e-5	**32.24 ± 4.72**	**91.92 ± 8.01**	**0.37 ± 0.05**

Table 2: Results on the NEAT evaluation routes for a fully supervised sensorimotor agent (AIM) trained with various adhoc techniques

0.5.5.2 Perfomance of the Learning Approaches

For all our semi-supervised and self-supervised learning approaches, we consider demonstrations from Town 1 and Town 2, as our small dataset $\mathcal{D}$ with ground-truth labels. We consider driving scenes from rest of the towns as our unlabeled dataset $\mathcal{U}_{\text{CARLA}}$. We also form an unlabeled dataset $\mathcal{U}_{\text{YouTube}}$ containing driving scenes from YouTube videos and combining these two dataset we have the unlabeled dataset $\mathcal{U}_{\text{CARLA + YouTube}}$. We consider various approaches for leveraging the unlabeled CARLA and YouTube data. To emphasize generalization, we do not pseudo-label the unseen test datasets or incorporate their unlabeled samples into the self-training in any way. We incorporate the same adhoc techniques in the self-training that improved performance in full supervised training. We report the closed-loop evaluation performance on Town05 long routes for all tests in Table 3.

Baselines: The first two rows of Table 3 shows evaluation scores of the AIM baseline trained supervisedly on Town 1 and 2. We report evaluation on two models with same internal specification and only differ in the outputs — one predicts the waypoints and the other one predicts confidence score in addition to the waypoints.

SelfD: Table 3 also shows results of SelfD with or without "What If" augmentation and confidence prediction in all combinations, for two unlabeled datasets $\mathcal{U}_{\text{CARLA}}$ and $\mathcal{U}_{\text{CARLA + YouTube}}$. But, we see no noticeable improvement and even reduction in perfomance in the evaluations compared to supervised training on Town1&2. SelfD training without "What If" augmentation on $\mathcal{U}_{\text{CARLA}}$ dataset shows very small improvement in the route completion, but this can also be accounted for noise in the evaluation. Thus, contrary to the claim by the authors of SelfD [69], we observe that SelfD gains no generalization with "What If" augmentation. Figure 7 demonstrates some of the cases where SelfD fails to produce appropriate pseduo-waypoints, but SemiD produces the correct ones.

Datasets	**Method**	**DS** (↑)	**RC** (↑)	**IS** (↑)
Labeled Dataset: $\mathcal{D}$	AIM (w/o confidence)	7.83 ± 0.56	51.01 ± 4.68	0.32 ± 0.03
	AIM (w/ confidence)	5.97 ± 0.78	54.35 ± 5.88	0.19 ± 0.02
Unlabeled Datasets: $\mathcal{U}_{\text{CARLA}}$	SelfD (w/o What If, w/o confidence)	6.07 ± 0.41	58.36 ± 3.08	0.17 ± 0.02
Labeled Dataset: $\mathcal{D}$	SelfD (w/o What If, w/ confidence)	8.3 ± 2.75	59.89 ± 9.87	0.22 ± 0.03
	SelfD (w/ What If, w/o confidence)	5.02 ± 0.51	53.32 ± 3.98	0.18 ± 0.01
	SelfD (w/ What If, w/ confidence)	4.67 ± 0.86	49.15 ± 6.43	0.15 ± 0.02
	SemiD (transductive training)	**13.25 ± 2.55**	**95.06 ± 5.33**	**0.13 ± 0.02**
	SemiD (finetuned)	12.58 ± 0.32	90.88 ± 3.91	0.14 ± 0.02
Unlabeled Datasets: $\mathcal{U}_{\text{CARLA + YouTube}}$	SelfD (w/o What If, w/o confidence)	6.66 ± 1.98	51.44 ± 7.23	0.23 ± 0.03
Labeled Dataset: $\mathcal{D}$	SelfD (w/o What If, w/ confidence)	5.17 ± 1.59	55.46 ± 8.57	0.16 ± 0.05
	SelfD (w/ What If, w/o confidence)	7.28 ± 0.65	47.21 ± 5.03	0.26 ± 0.02
	SelfD (w/ What If, w/ confidence)	6.22 ± 1.12	51.34 ± 6.8	0.21 ± 0.03
	SemiD (transductive training)	8.87 ± 2.01	92.93 ± 6.77	0.11 ± 0.03
	SemiD (finetuned)	10.15 ± 1.23	93.38 ± 6.31	0.1 ± 0.01
	OVRL	7.78 ± 1.26	83.13 ± 7.78	0.12 ± 0.04
	OVRL + SemiD (transductive training)	5.51 ± 2.26	**97.54 ± 2.74**	0.06 ± 0.02
	OVRL + SemiD (finetuned)	9.79 ± 1.22	90.66 ± 0.29	0.14 ± 0.01
Unlabeled Datasets: $\mathcal{U}_{\text{YouTube}}$	SemiD (transductive training)	-	-	-
Labeled Dataset: $\mathcal{D}$	SemiD (finetuned)	-	-	-

Table 3: Results on the NEAT evaluation routes for sensorimotor agent (AIM) trained with various semi-supervised and self-supervised learning approaches

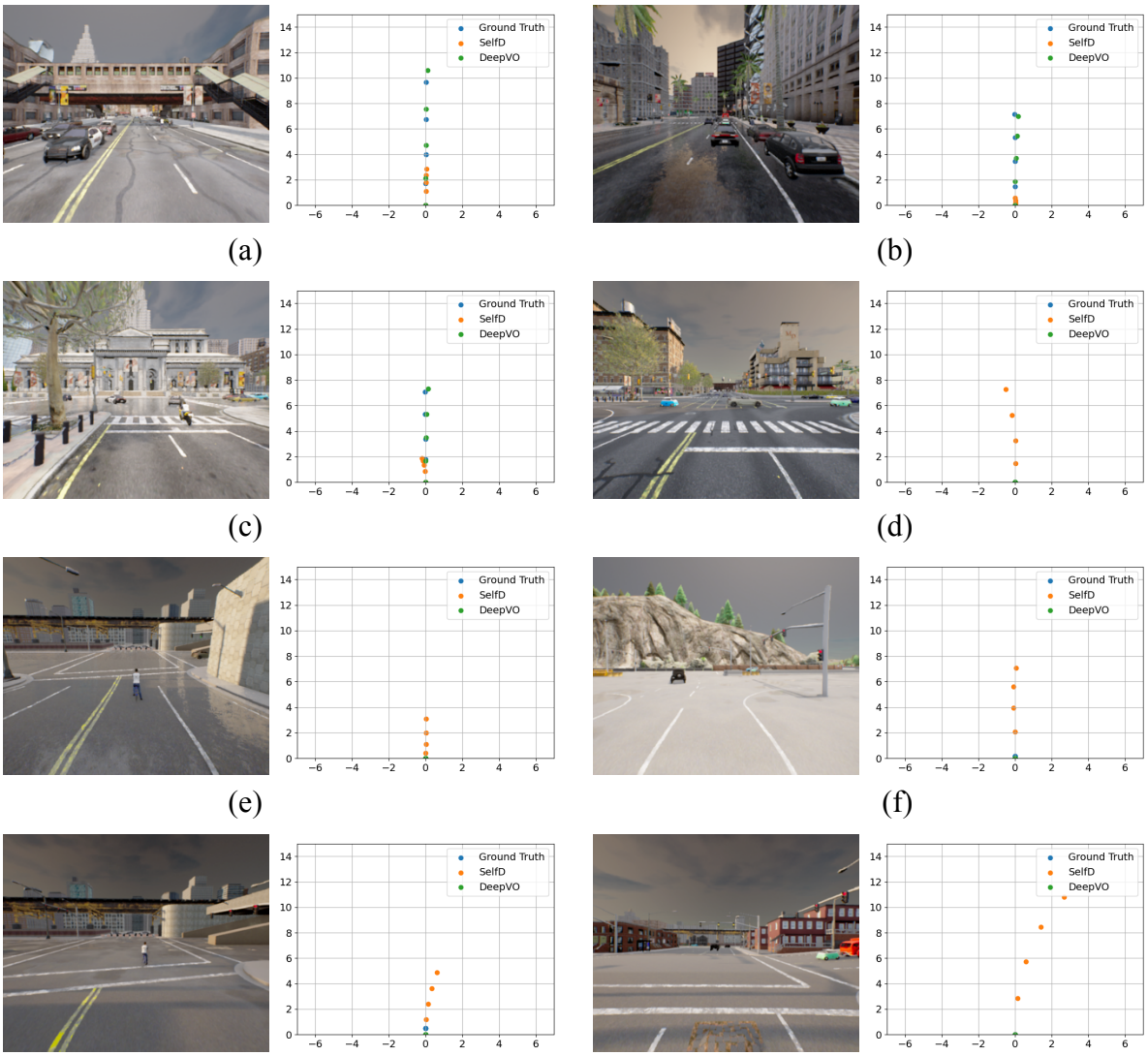

Figure 7: Some failure cases where SelfD could not recover the desired pseduo-waypoints but correctly recovered by DeepVO. Examples (a), (b), (c), show some situations where the ego-vehicle should moves forward but SelfD predicts waypoints to slow down the ego-vehicle. Examples (d), (e) show situations where ego-vehicle should not move because of the cyclist infront or the crossing vehicles, but in this case SelfD predicts waypoints to keep the ego-vehicle moving. Examples (f), (g), (h), show situations where the ego-vehicle should stop because of the red-light, yet SelfD predicts waypoints violating the red-light stop.

SemiD: We train our DeepVo architechture with Town1&2 data containing driving scenes and its corresponding ego-vehicle positions and gather pseduo-waypoints and pseudo-target-point for all the unlabeled dataset. We report results of SemiD for three unlabled dataset $\mathcal{U}_{\text{CARLA}}$, $\mathcal{U}_{\text{YouTube}}$ and $\mathcal{U}_{\text{CARLA + YouTube}}$ in Table 3. Contrary to the claim by the authors of SelfD [69], we observe this method improve driving perfomance by a huge margin. With only transductive learning stage (i.e. training with pseduo-labels) of semi-supervised learning, we achieve a near doubling score in driving score (DS) and route completion (RC), from 7.83 to 13.25 in driving score and 51% to 95% in route completion. We observe this performance improvement across all unlabeled datasets. We note that while our supervised AIM Baseline benefits from the known fixed perspective transform assumption, SemiD does not incorporate any knowledge of camera parameters and must learn it from the data, yet SemiD achieves comparable perfomance in route completion as fully supervisedly trained model on all CARLA training towns (see Table 2).

OVRL: We also shows perfomance of the self-supervised learning approach OVRL Table 3. We pre-train the ResNet34 backbone on the $\mathcal{U}_{\text{CARLA + YouTube}}$ unlabeled dataset self-supervisedly and fine-tuned on Town1&2. Here also we gain improvement in driving score from 51% to 83%. We also combine OVRL and SemiD, by using the pretrained ResNet backbone from OVRL in SemiD. In this case, we achieve further improvement in driving performance (97.5% in RC with less variance) which surpasses the avgerage route completion achieved by supervisedly trained model on all CARLA training towns and after finetuning on the Town1&2, we achieve better infraction score and thus better driving score.

Limitations: Even though, in all of these learning approaches we achieve near perfect route completion, we can observe that we do not gain improvement in infraction score. The reason is two fold. Firstly, since the driving agent now covers on average more than 90% of the routes, the total number of infractions also increases compared to when the agent covers 50% of the routes (as for AIM Baseline) and because of the multiplicative nature of the infraction score metric, the infraction score and driving score is very much affected. Secondly, in both SelfD and SemiD, the generated pseduo-waypoints are noisy especially when the ego-vehicle stands still and therefore it is hard for the the model map appropriate behavior in all infraction scenarios and for OVRL it happens because the traffic light signal in the european towns is significantly different from the evaluation town which has the highway traffic lights. Beyond these reasons, infractions due to invisibilty (e.g. the highway traffic light is not visible or ego-vehicle changes lane but the rear vehicle collide with it or the passenger starts walking after the car already have gone past it) is one of the biggest reason why infraction score is affected even in supervised training.

0.6 Conclusion

In this thesis, we proposed, SemiD, a novel semi-supervised learning approach based on deep visual odometry. We have also explored some prior works in our autonomous driving framework that use semi-supervised and self-supervised learning for navigation, namely 'SelfD' and 'OVRL'. We showed that SemiD outperforms the prior works on the challenging NEAT evaluation routes by achieving 95% in RC. We also observed that OVRL can learn generalizable representations and bring substantial improvement in driving perfomance with 97.5% in RC when combined with SemiD. Thus, we showed that we can significantly improve perfomance of a self-driving agent without incurring additional data collection or annotation efforts, i.e., for a new platform, perspective, use-case, or ambient settings. We also showed that our model architechture can learn how to maneuver in diverse driving situations in different camera configurations. Thus, we envision that the proposed approach can be leveraged in real-world generalization and adaptation settings for autonomous driving systems.

Even though SemiD achieved almost perfect route completion, it incurred infraction frequently. Thus, a future direction could be to investigate this issue. Also our model development was restricted to large-scale training with single frame. Therefore it would be interesting to study if this method can gain benefit from temporal demonstrations. Finally, beyond complex 3D navigation, it would be interesting to explore the applicability of our proposed training framework for learning various embodied tasks from unlabeled web data.

Bibliography

[1] Michael Bain and Claude Sammut. "A Framework for Behavioural Cloning". In: 15 (Mar. 2000).

[2] Nicolas Ballas et al. "Delving Deeper into Convolutional Networks for Learning Video Representations". In: (Nov. 2015). arXiv: 1511.06432 [cs.CV].

[3] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst". In: (Dec. 2018). arXiv: 1812.03079 [cs.RO].

[4] Aseem Behl et al. "Label Efficient Visual Abstractions for Autonomous Driving". In: (May 2020). arXiv: 2005.10091 [cs.CV].

[5] RICHARD BELLMAN. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961. ISBN: 9780691079011. URL: http://www.jstor.org/stable/j.ctt183ph6v (visited on 09/10/2022).

[6] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: (Apr. 2016). arXiv: 1604.07316 [cs.CV].

[7] Andreas Bühler et al. "Driving Through Ghosts: Behavioral Cloning with False Positives". In: (Aug. 2020). arXiv: 2008.12969 [cs.CV].

[8] Benjamin Caine et al. "Pseudo-labeling for Scalable 3D Object Detection". In: (Mar. 2021). arXiv: 2103.02093 [cs.CV].

[9] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: (Apr. 2021). arXiv: 2104.14294 [cs.CV].

[10] Mathilde Caron et al. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: (June 2020). arXiv: 2006.09882 [cs.CV].

[11] Sergio Casas, Abbas Sadat, and Raquel Urtasun. "MP3: A Unified Model to Map, Perceive, Predict and Plan". In: (Jan. 2021). arXiv: 2101.06806 [cs.RO].

[12] Matthew Chang, Arjun Gupta, and Saurabh Gupta. "Semantic Visual Navigation by Watching YouTube Videos". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 4283–4294. URL: https://proceedings.neurips.cc/paper/2020/file/2cd4e8a2ce081c3d7c32c3cde4312ef7-Paper.pdf.

[13] Chenyi Chen et al. "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2722–2730. DOI: 10.1109/ICCV.2015.312.

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

[15] Dian Chen et al. "Learning by Cheating". In: (Dec. 2019). arXiv: 1912.12294 [cs.RO].

[16] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: (Feb. 2020). arXiv: 2002.05709 [cs.LG].

[17] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning". In: (Nov. 2020). arXiv: 2011.10566 [cs.CV].

[18] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. "NEAT: Neural Attention Fields for End-to-End Autonomous Driving". In: (Sept. 2021). arXiv: 2109.04456 [cs.CV].

[19] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (June 2014). arXiv: 1406.1078 [cs.CL].

[20] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: (Dec. 2014). arXiv: 1412.3555 [cs.NE].

[21] Felipe Codevilla et al. "End-to-end Driving via Conditional Imitation Learning". In: (Oct. 2017). arXiv: 1710.02410 [cs.RO].

[22] Felipe Codevilla et al. "Exploring the Limitations of Behavior Cloning for Autonomous Driving". In: (Apr. 2019). arXiv: 1904.08980 [cs.CV].

[23] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: (Nov. 2017). arXiv: 1711.03938 [cs.LG].

[24] Angelos Filos et al. "Can Autonomous Vehicles Identify, Recover From, and Adapt to Distribution Shifts?" In: (June 2020). arXiv: 2006.14911 [cs.LG].

[25] Philipp Fischer et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: (Apr. 2015). arXiv: 1504.06852 [cs.CV].

[26] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 3354–3361.

[27] Priya Goyal et al. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour". In: (June 2017). arXiv: 1706.02677 [cs.CV].

[28] Jean-Bastien Grill et al. “Bootstrap your own latent: A new approach to self-supervised Learning”. In: (June 2020). arXiv: 2006.07733 [cs.LG].

[29] Saurabh Gupta et al. “Cognitive Mapping and Planning for Visual Navigation”. In: (Feb. 2017). arXiv: 1702.03920 [cs.CV].

[30] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. “Causal Confusion in Imitation Learning”. In: (May 2019). arXiv: 1905.11979 [cs.LG].

[31] Jeffrey Hawke et al. “Urban Driving with Conditional Imitation Learning”. In: (Nov. 2019). arXiv: 1912.00177 [cs.CV].

[32] Kaiming He et al. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: (Nov. 2019). arXiv: 1911.05722 [cs.CV].

[33] Joel Janai et al. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. Vol. 12. 1-3. Foundations, Trends in Computer Graphics, and Vision, 2020.

[34] Ilya Kostrikov, Denis Yarats, and Rob Fergus. “Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels”. In: (Apr. 2020). arXiv: 2004.13649 [cs.LG].

[35] Michael Laskin et al. “Reinforcement Learning with Augmented Data”. In: (Apr. 2020). arXiv: 2004.14990 [cs.LG].

[36] Yann Lecun et al. “Off-Road Obstacle Avoidance through End-to-End Learning.” In: Jan. 2005.

[37] Dong-Hyun Lee. “Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks”. In: *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (July 2013).

[38] Guohao Li et al. “OIL: Observational Imitation Learning”. In: (Mar. 2018). arXiv: 1803.01129 [cs.CV].

[39] Xiaodan Liang et al. “CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving”. In: (July 2018). arXiv: 1807.03776 [cs.CV].

[40] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: (Nov. 2017). arXiv: 1711.05101 [cs.LG].

[41] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: (Aug. 2016). arXiv: 1608.03983 [cs.LG].

[42] Lina Mezghani et al. “Memory-Augmented Reinforcement Learning for Image-Goal Navigation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022* (Jan. 2021). arXiv: 2101.05181 [cs.CV].

[43] Urs Muller et al. “Off-road obstacle avoidance through end-to-end learning”. In: *NeurIPS*. 2006.

[44] Matthias Müller et al. “Driving Policy Transfer via Modularity and Abstraction”. In: (Apr. 2018). arXiv: 1804.09364 [cs.RO].

[45] E. Ohn-Bar et al. “Learning Situational Driving”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2020, pp. 11293–11302. DOI: 10.1109/CVPR42600.2020.01131. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01131.

[46] Takayuki Osa et al. “An Algorithmic Perspective on Imitation Learning”. In: (Nov. 2018). DOI: 10.1561/2300000053. arXiv: 1811.06711 [cs.RO].

[47] Dean A. Pomerleau. “ALVINN: An Autonomous Land Vehicle in a Neural Network”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 1. Morgan-Kaufmann, 1988. URL: https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[48] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. “Multi-Modal Fusion Transformer for End-to-End Autonomous Driving”. In: (Apr. 2021). arXiv: 2104.09224 [cs.CV].

[49] Aditya Prakash et al. “Exploring Data Aggregation in Policy Learning for Vision-Based Urban Autonomous Driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[50] Mamshad Nayeem Rizve et al. “In Defense of Pseudo-Labeling: An Uncertainty-Aware Pseudo-label Selection Framework for Semi-Supervised Learning”. In: (Jan. 2021). arXiv: 2101.06329 [cs.LG].

[51] Axel Sauer, Nikolay Savinov, and Andreas Geiger. “Conditional Affordance Learning for Driving in Urban Environments”. In: (June 2018). arXiv: 1806.06498 [cs.RO].

[52] Ardi Tampuu et al. “A Survey of End-to-End Driving: Architectures and Training Methods”. In: *IEEE Transactions on Neural Networks and Learning Systems, 2020* (Mar. 2020). DOI: 10.1109/TNNLS.2020.3043505. arXiv: 2003.06404 [cs.AI].

[53] Faraz Torabi, Garrett Warnell, and Peter Stone. “Behavioral Cloning from Observation”. In: (May 2018). arXiv: 1805.01954 [cs.AI].

[54] Faraz Torabi, Garrett Warnell, and Peter Stone. “Recent Advances in Imitation Learning from Observation”. In: (May 2019). arXiv: 1905.13566 [cs.RO].

[55] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. “End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances”. In: (Nov. 2019). arXiv: 1911.10868 [cs.LG].

[56] Dequan Wang et al. “Monocular plan view networks for autonomous driving”. In: *IROS*. 2019.

[57] Jingke Wang et al. “Learning hierarchical behavior and motion planning for autonomous driving”. In: (May 2020). arXiv: 2005.03863 [cs.RO].

[58] Sen Wang et al. “DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks”. In: (Sept. 2017). DOI: 10.1109/ICRA.2017.7989236. arXiv: 1709.08429 [cs.CV].

[59] Benjamin Wilson et al. “Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*. 2021.

[60] Zhirong Wu et al. “Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination”. In: (May 2018). arXiv: 1805.01978 [cs.CV].

[61] Yi Xiao et al. “Action-Based Representation Learning for Autonomous Driving”. In: (Aug. 2020). arXiv: 2008.09417 [cs.CV].

[62] Yi Xiao et al. “Multimodal End-to-End Autonomous Driving”. In: (June 2019). DOI: 10.1109/TITS.2020.3013234. arXiv: 1906.03199 [cs.CV].

[63] Karmesh Yadav et al. “Offline Visual Representation Learning for Embodied Navigation”. In: (Apr. 2022). arXiv: 2204.13226 [cs.CV].

[64] Jihan Yang et al. “ST3D: Self-training for Unsupervised Domain Adaptation on 3D Object Detection”. In: (Mar. 2021). arXiv: 2103.05346 [cs.CV].

[65] Denis Yarats et al. “Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning”. In: (July 2021). arXiv: 2107.09645 [cs.AI].

[66] Jure Zbontar et al. “Barlow Twins: Self-Supervised Learning via Redundancy Reduction”. In: (Mar. 2021). arXiv: 2103.03230 [cs.CV].

[67] Guangyao Zhai et al. “PoseConvGRU: A Monocular Approach for Visual Ego-motion Estimation by Learning”. In: (June 2019). arXiv: 1906.08095 [cs.CV].

[68] Jimuyang Zhang and Eshed Ohn-Bar. “Learning by Watching”. In: (June 2021). arXiv: 2106.05966 [cs.CV].

[69] Jimuyang Zhang, Ruizhao Zhu, and Eshed Ohn-Bar. “SelfD: Self-Learning Large-Scale Driving Policies From the Web”. In: (Apr. 2022). arXiv: 2204.10320 [cs.CV].

[70] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. “Action-Conditioned Contrastive Policy Pretraining”. In: (Apr. 2022). arXiv: 2204.02393 [cs.CV].

[71] Albert Zhao et al. “LaTeS: Latent Space Distillation for Teacher-Student Driving Policy Learning”. In: *CoRR* abs/1912.02973 (2019). URL: http://arxiv.org/abs/1912.02973.

[72] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. “Does computer vision matter for action?” In: *Science Robotics 22 May 2019: Vol. 4, Issue 30, eaaw6661* (May 2019). DOI: 10.1126/scirobotics.aaw6661. arXiv: 1905.12887 [cs.CV].