



Masterarbeit

Exploring Semi-supervised and Self-supervised Learning Approaches in Autonomous Driving

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Partha Ghosh, partha.ghosh@student.uni-tuebingen.de, 2022

Bearbeitungszeitraum: 01/05/2022 - 01/11/2022

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

Selbstständigkeitserklärung

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

Partha Ghosh (Matrikelnummer 5620076), October 26, 2022

Abstract

How can we learn generalized autonomous driving systems for robust vision-based navigation in complex and dynamic environments? The current status of solving these visual navigation tasks is to train visual representations and navigation policies with direct supervision. Yet, vast amounts of unlabeled, highly diverse ego-centric navigation data are freely available on the internet. Therefore, in this work, we study several different approaches to effectively utilize this huge diverse navigational data to robustly scale across perspectives, scenarios, environmental conditions, platforms, and geographical locations. We study two existing works in this field that employ semi-supervised and self-supervised learning, namely ‘SelfD’ [68] and ‘OVRL’ [61]. Moreover, in this work, we introduce SemiD, a framework for learning scalable driving by utilizing massive amounts of online monocular driving videos. Our key idea is to leverage deep visual odometry for iterative semi-supervised training when learning imitation learning agents from unlabeled data. To handle various driving scenes, unconstrained viewpoints, 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 pseudo-labels for the unlabeled data and augment the decision-making knowledge and robustness of the model via semi-supervised training. We use a huge dataset of publicly available YouTube videos to train SemiD and comprehensively analyze its generalization benefits across challenging CARLA evaluation routes. 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

Acknowledgments

I want to thank Bernhard Jaeger and Katrin Renz for supervising my thesis 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

1	Introduction	11
2	Related Work	13
2.1	End-to-End Autonomous Driving	13
2.2	Imitation Learning	13
2.3	Semi-Supervised Learning for Navigation	14
2.3.1	Semi-supervised Learning	14
2.3.2	Learning by Cheating	14
2.4	Self-Supervised Visual Representation Learning	15
2.4.1	Self-supervised learning	15
2.4.2	Contrastive Learning	15
2.4.3	Non-Contrastive Learning	15
2.4.4	Action-CConditioned Policy Pre-Training (ACO)	16
3	Autonomous Driving Framework	17
3.1	Imitation Learning	17
3.2	Input and Output Representations	18
3.2.1	Input Representation	18
3.2.1.1	Driving Scenes	18
3.2.1.2	Navigational Planner	18
3.2.2	Output Representation	18
3.3	Waypoint Prediction Network: AIM	20
3.4	PID Controller	20
3.5	Loss Function	21
4	Learning Approaches	23
4.1	SemiD: Semi-supervised Driving with Deep Visual Odometry	23
4.1.1	Deep Visual Odometry	24
4.1.1.1	Feature-encoding Module	24
4.1.1.2	Memory-propagating Module	25
4.1.2	Generating Pseudo-labels for Video frames	25
4.1.3	Model Pre-Training and Fine-Tuning	27
4.2	SelfD: Self-Learning Large-Scale Driving Policies	29
4.2.1	Conditional Imitation Learning from Observations	29

Contents

4.2.2	Initial Data Assumption	29
4.2.3	Self-supervised Training Process	30
4.2.4	BEV Plan Network	30
4.2.5	“What If” Pseudo-Labeling of the Unlabeled Data	30
4.2.6	Model Pre-Training and Fine-Tuning	31
4.3	OVRL: Offline Visual Representation Learning	31
4.3.1	Self-supervised Pre-Training	32
4.3.2	Implementation Details	32
4.3.3	Downstream Learning	34
5	Experimental Results	35
5.1	Task	35
5.2	Implementation Details	36
5.2.1	Data Cleansing	36
5.2.2	Stratified Sampling	36
5.2.3	Image Augmentation	37
5.3	Dataset	37
5.4	Evaluation Metrics	38
5.4.1	Route Completion (RC)	38
5.4.2	Infraction Score (IS)	38
5.4.3	Driving Score (DS)	39
5.5	Results	39
5.5.1	Performance of Supervised Training	39
5.5.2	Performance of the Learning Approaches	40
6	Conclusion	45

1 Introduction

How can we teach autonomous systems to drive based on visual input? How can we learn generalized autonomous driving models for robust vision-based navigation in complex and dynamic environments? While humans can effortlessly transfer general navigation knowledge across different environments and settings (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 strategies 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 current status of 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 [16, 21]. The agent is given waypoints generated by an expert driver, along with the expert’s sensory input (e.g. driving scene). The goal of learning is to produce a policy that will mimic the expert’s actions given the corresponding sensory input [1, 7, 14, 15, 29, 31, 38, 39, 43, 35, 45, 46, 47, 67]. Now, every minute, a 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 have been shown to be generalizable across visual tasks ranging from image classification, semantic segmentation, to object detection [17, 28, 32, 11, 10]. However, these methods are primarily built for learning features for recognition tasks rather than navigation. Therefore, in this work, we aim to effectively utilize such freely available demonstration data to improve the efficiency, safety, and scalability of generalized real-world navigation agents.

We explore two different types 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 different. In semi-supervised learning, we devise strategies to generate reasonable pseudo-labels to the unlabeled input driving scenes so that upon training with these pseudo-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 representations 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 categorized 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. [68] in our autonomous driving framework. We explore in our self-driving framework, “OVRL”, a self-supervised learning approach proposed by Yadav et. al. [61] 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 CARLA Town05 long evaluation routes. Also, we propose a data cleansing and sampling technique for effective training. We show that combining the aforementioned technique with mild image augmentation improves the route completion by a huge margin and therefore also improves the driving performance.

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 CARLA 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 the training process to improve driving performance.
- We find that image augmentations are quite important for achieving good performance.
- We show that the inertia problem can be tackled 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 2. Then, in Section 3, we introduce our autonomous driving framework where we will deploy all our learning approaches. In Section 4, we describe in detail various semi-supervised and self-supervised learning approaches. Next, we discuss the experiment results in Section 5, followed by the conclusion in Section 6.

2 Related Work

2.1 End-to-End Autonomous Driving

End-to-End autonomous driving describes approaches in which the entire driving task is done by a single neural network that directly maps the raw sensory input 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 advantage 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. in their ALVINN-vehicle [46] and still, active research is going on in this field [33, 51]. Imitation Learning for driving has advanced significantly [7, 21, 43] and is currently employed in several state-of-the-art approaches, some of which predict waypoints [12, 16, 25], whereas others directly predict vehicular control [4, 8, 22, 44, 60, 48]. While other learning-based driving methods such as affordances [50, 59] and reinforcement learning [15, 53, 55] could also benefit from semi-supervised or self-supervised learning, in this work, we try to improve imitation learning-based autonomous driving through semi-supervised and self-supervised learning.

2.2 Imitation Learning

Imitation Learning is one of the most promising approaches for self-driving. Our main idea is to leverage the scale and diversity of readily accessible online ego-centric navigation data to learn a robust conditional imitation learning policy [16, 21]. Even though learning from labeled demonstrations can significantly simplify the challenging vision-based policy learning task [1, 7, 14, 15, 29, 31, 38, 39, 43, 35, 45, 46, 47, 67, 70, 71], observed images in our settings are not labeled with the corresponding actions of a demonstrator. Therefore we aim to generalize current conditional imitation learning (CIL) approaches [16, 21, 22] to learn from unlabeled image observations such that the agent can drive in complex urban scenarios. To address this challenging observational learning task, prior work has recently explored introducing different restrictive assumptions, including an interactive environment for on-policy data collection [6], access to a hand-designed reward function [13], or

demonstrator optimality [6, 52]. We instead facilitate scalable training from diverse data sources, by employing semi-supervised or self-supervised learning approaches. Also, our resulting model can be used to bootstrap other methods for policy training, e.g., model-free or model-based reinforcement learning approaches [15, 39, 44, 53].

2.3 Semi-Supervised Learning for Navigation

2.3.1 Semi-supervised Learning

Semi-supervised learning is a machine learning technique that uses a small amount of labeled data together with a large amount of unlabeled data during training. It falls between supervised learning (with only labeled training data) and unsupervised learning (with no labeled training data). It is a special instance of weak supervision. Semi-supervised learning 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 labeling 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.

2.3.2 Learning by Cheating

The prior works [9, 62, 37, 49], emphasizing semi-supervised learning through image and object-level recognition tasks, have 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, 47]. This issue has been addressed in the recent work ‘Learning by Cheating’ (LBC) by Chen et al. [16]. LBC utilizes a multi-stage training step in which privileged (i.e. teacher) sensorimotor CIL agent is used 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 have access to extensive ground truth information through the simulation, it produces highly plausible and clean trajectories for the driving scenes, 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 SemiD (our semi-supervised

driving approach using deep visual odometry, see Section 4.1) we generate pseudo-labels for diverse out-of-distribution driving scenes to enable transductive learning of the sensorimotor agent. On the other hand, SelfD (a semi-supervised driving approach that we study in this thesis, see Section 4.2) leverages the same visuomotor architecture as teacher and student. We also train in an inherently noisy settings, as the teacher inference is performed on a diverse out-of-distribution image data and not on the ground truth training dataset.

2.4 Self-Supervised Visual Representation Learning

2.4.1 Self-supervised learning

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

2.4.2 Contrastive Learning

The core idea of contrastive learning is to attract the positive example pairs and repulse the negative example pairs (where the positive pair indicates two different views of the same image whereas the negative pair indicates views of two different images). This methodology has been recently popularized for self-supervised representation learning [58]. Siamese networks have been developed using simple and effective instantiations of contrastive learning [32, 17, 65]. In practice, these contrastive learning methods benefit from a large number of negative samples. These samples are maintained in a memory bank [58]. In a Siamese network, MoCo [32] maintains a queue of negative samples and use a momentum encoder to improve the consistency of the queue. SimCLR [17] directly uses negative samples in the batch but requires a large batch size to work well.

2.4.3 Non-Contrastive Learning

Recent works have shown that one 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 from a momentum encoder. Methods inspired by BYOL such as SimSiam, DINO etc. [18, 10], have shown that this method works even without a momentum encoder. In particular,

we will concentrate on the method DINO in this thesis and describe in detail its internal mechanisms (see Section 4.3.1).

2.4.4 Action-COnditioned Policy Pre-Training (ACO)

Action-Conditioned policy pre-training paradigm, uses contrastive learning to capture important features in the neural representation which are relevant to the decision-making and benefits downstream tasks. The method, proposed by Zhang et. al. [69] 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 pre-trained 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 these action-conditioned contrastive pairs, the representation captures many policy-related features that are highly correlated to the actions. The experimental results show that ACO successfully learns generalizable features which can be then used for the downstream task such as policy learning through Imitation Learning (IL) and Reinforcement Learning (RL) in end-to-end driving, and Lane Detection (LD).

In contrast, we focus on non-contrastive learning, particularly, the method DINO proposed by Caron et. al. [10], to learn visual representations from the unlabeled data. These generic representations can then be transferred to the policy learning task via imitation learning.

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 handle 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 behavior rather than to specify a reward function.

3.1 Imitation Learning

The goal of Imitation Learning (IL) is, for an agent to learn a policy π_θ , 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 brake) or a desired future trajectory relative to the ego-vehicle, i.e., a set of K waypoints [16, 43] in the BEV (birds-eye-view) space. In the latter case, future waypoints can be paired with a hand-designed or learned motion controller to produce the low-level action [16, 43]. In this work, we focus on the latter representation because of its interpretability and generalizability. To find the mapping, we consider the Behavior Cloning (BC) approach of IL. To explore different learning approaches in Section 4, we would need access to a 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 target 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 target location. 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 $\pi_\theta : \mathcal{X} \rightarrow \mathcal{W}$ with learnable parameters $\theta \in \mathbb{R}^d$. In BC, the policy π_θ is learned by training a neural network in a supervised manner using the dataset, \mathcal{D} , with a loss function, \mathcal{L} i.e.

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

We use the L_1 distance between the predicted trajectory, $\pi_\theta(\mathbf{X})$, 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 determined as $\mathbf{A} = \mathbb{I}(\mathbf{W})$.

3.2 Input and Output Representations

3.2.1 Input Representation

3.2.1.1 Driving Scenes

We note that, even though our collected expert demonstrations 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 self-driving have shown that using observation histories may not lead to performance improvement [30, 36, 3, 54]. 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 pixels which we resize and crop to 256×256 to remove radial distortion at the edges. Figure 3.1 shows some example driving scenes from CARLA and YouTube driving videos.

3.2.1.2 Navigational Planner

We use the same standard protocol of CARLA 0.9.10 and assume that high-level target 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 target locations are sparse and can be very far away as opposed to the local waypoints predicted by the policy π_θ .

3.2.2 Output Representation

We predict the waypoints \mathbf{W} of the ego-vehicle in BEV space, centered at the current coordinate frame of the ego-vehicle. The waypoints are represented by a sequence of 2D coordinates, $\mathbf{W} = \{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^T$. Following [47], we use $T = 4$, which is the default number of waypoints required by the inverse dynamics model in the author's implementation.

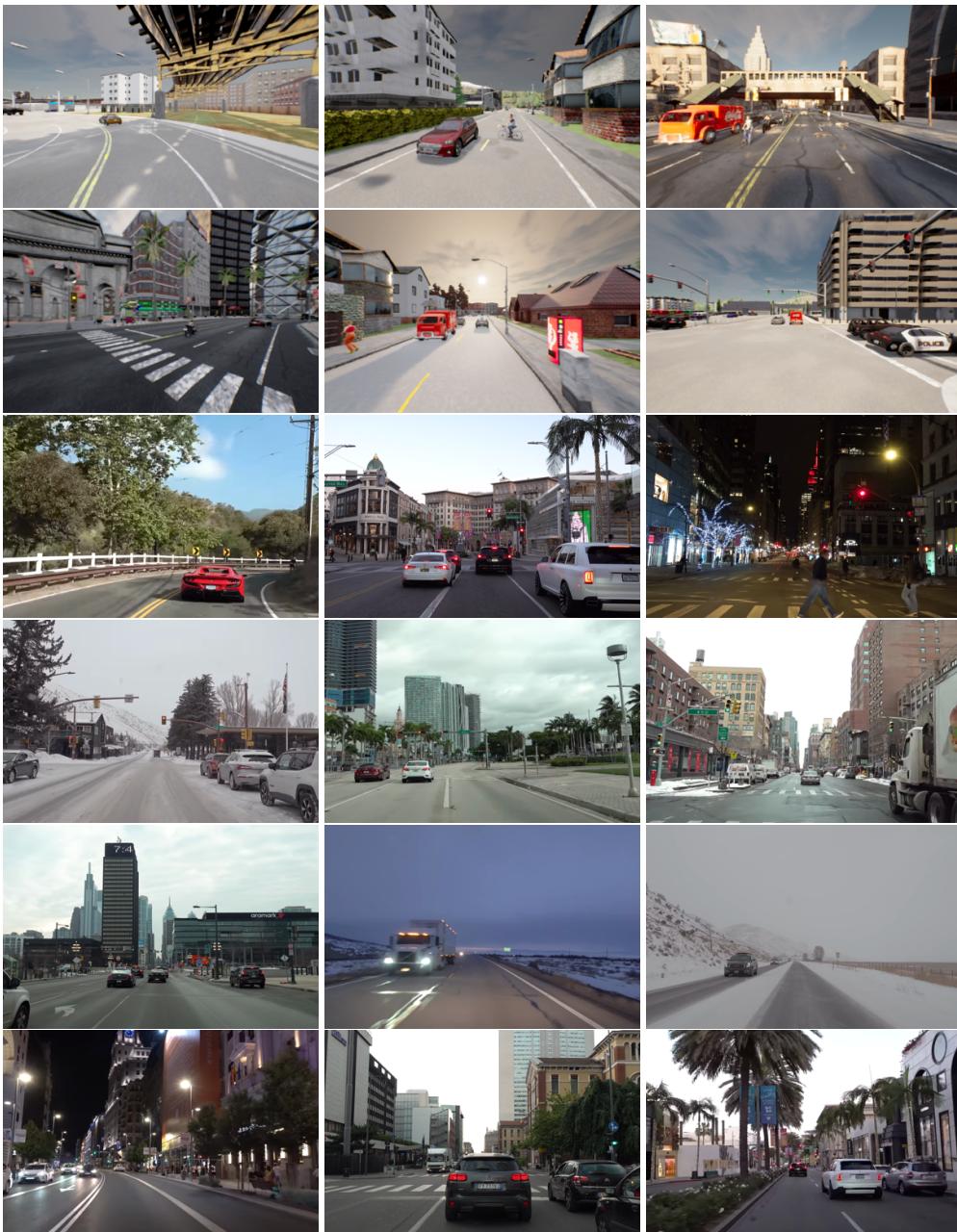


Figure 3.1: First two rows shows examples of driving scenes from the CARLA simulator and the rest of the images are extracted from the YouTube driving videos.

3.3 Waypoint Prediction Network: AIM

We adopt the image-based baseline network architecture ‘AIM’ proposed by Prakash et. al. [47]. As shown in Figure 3.2, first, we encode the input driving scene with a ResNet34 architecture into a 512-dimensional feature-encoding. We then pass this 512-dimensional feature vector through an MLP (comprised of 2 hidden layers of size 256 and 128 respectively) to reduce its dimensionality to 64 for computational efficiency. After that, we feed it to the auto-regressive module of the waypoint prediction network implemented using GRUs [19]. The hidden state of the GRU is initialized with a 64-dimensional zero vector. The update gate of the GRU manages the flow of information encoded in the hidden state to the output and carry over to the next time-step. Also, the GRU takes as input the current position of the ego-vehicle and the target location (Section 3.1), which allows it to focus on the relevant context in the hidden state to predict the next waypoint. As the GPS coordinates of the target location (transformed to the ego-vehicle coordinate frame) lies in the same BEV space as the predicted waypoints, we provide it as input to the GRU rather than to the encoder and thus it correlates better compared to representing the target location in the perspective image domain [16]. Following [25], we use a single-layered GRU followed by a linear layer which takes in the hidden state and predicts the differential waypoints $\{\delta w_t\}_{t=1}^T$ for $T = 4$ future time-steps in the ego-vehicle’s current coordinate frame. Therefore, the predicted future waypoints are given by $\{w_t = w_{t-1} + \delta w_t\}_{t=1}^T$. In the first time step, the input to the GRU is provided as $(0, 0)$ since the BEV space is centered at the ego-vehicle’s position.

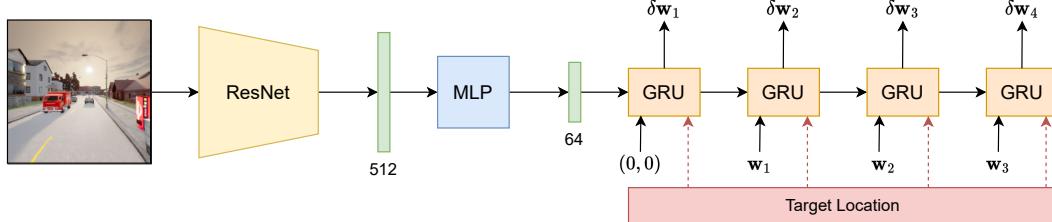


Figure 3.2: Architecture of the waypoint prediction network AIM [47]

3.4 PID Controller

We use two PID controllers for the lateral and longitudinal vehicular controls i.e. to obtain steer, throttle and brake values from the predicted waypoints, $\{w_t\}_{t=1}^T$. First we compute the vectors between the waypoints of consecutive time-steps. The longitudinal controller then takes as input the magnitude of a weighted average of these vectors whereas the lateral controller takes as input their orientation. For the PID controllers,

we use the same configuration as provided in the codebase of the author Chen et. al. [16].

3.5 Loss Function

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

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

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

4 Learning Approaches

Our goal is to train driving policies at scale to facilitate generalization across different environments and settings. Therefore, 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 SemiD, 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 by Zhang et. al. [68] and incorporate it into 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. [61] which was studied in the domain of embodied navigation.

4.1 SemiD: Semi-supervised Driving with Deep Visual Odometry

In this section, we will introduce SemiD, 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 [56, 66], a learning-based ego-motion estimation method, because of the following reasons:

- It is robust to image noise and independent of camera calibration [56, 66].
- In our expert driving dataset, the weather conditions and the time of the day change 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 pseudo-labels satisfy this criterion as the estimated waypoints with deep visual odometry are fairly accurate due to 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.

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 4.1: Configuration of the CNN

- 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 driving.
- Learning-based visual odometry method can learn the scale of the waypoints i.e. the predicted waypoints will be in the same unit as that of the ground truth waypoints; whereas for classical methods we have to delve into hyperparameters tuning to achieve the appropriate scale.

4.1.1 Deep Visual Odometry

We use an end-to-end learning approach following [56, 66] 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 Network. The feature-encoding module encodes the short-term motion features in a pair of images, while the memory-propagating module captures the long-term motion features in the consecutive image pairs.

4.1.1.1 Feature-encoding Module

In order to learn the geometric relationships from two adjacent images, we use the following CNN architecture, inspired by FlowNetSimple architecture [24] ignoring the decoder part of it and only focusing on the convolutional encoder. Contrary to DeepVO [56] and PoseConvGRU [66], which uses huge 10-layered CNN architectures, we use a very simple and lightweight 5-layered CNN architecture as shown in Table 4.1. 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 layer. We use maxpool of size 2 in the first 2 layers and

use maxpool of size 4 for the rest of the layers. The reason behind this is that having a 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 pooling layers in the later layers enforces the network to propagate only the relevant information for relative motion estimation and therefore also ignores the global context and also helps in reducing the number of learnable parameters as well. The input to the CNN is 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 map 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, as we have seen in our experiments that performing pre-processing operations to the images such as blurring, adding noise, random clipping, etc., worsen the relative motion estimates as the network cannot learn the geometric relationships between the objects in the image pair.

4.1.1.2 Memory-propagating Module

Following [66], we use a stacked GRU (Gated Recurrent Unit) [2] as our memory-propagating module as shown in Figure 4.1. The memory module builds a set of chronological visual representations from the CNN embeddings of the sequence of image pairs. Because of their ability of remembering histories, GRUs 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 units) 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 embedding 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 [25], we use a single-layered 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.

4.1.2 Generating Pseudo-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 3.1), which includes driving scenes and its corresponding ego-vehicle absolute positions, to learn the relative motion estimation. This

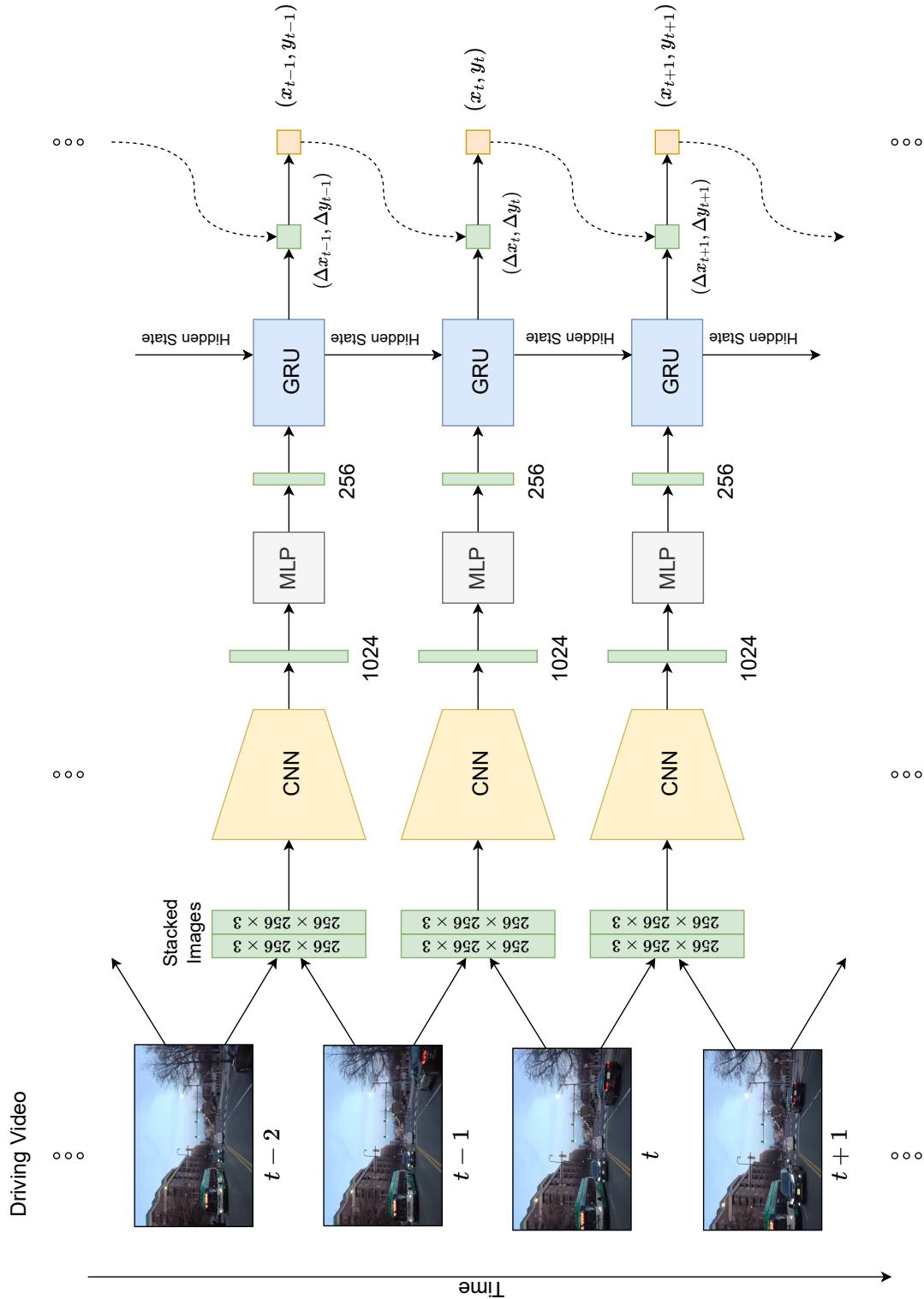


Figure 4.1: Architecture of the ConvGRU-based monocular VO system.

assumption is reasonable considering that there are several publicly available driving datasets containing ego-vehicle position and orientation for relative motion estimation [26, 57]. At 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 to 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 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 pseudo-labels i.e. pseudo-waypoints and pseudo-target-point for the initial driving scene. We repeat this procedure for all the unlabeled driving scenes and thus we create our pseudo-labeled dataset $\hat{\mathcal{D}}_{\text{SemiD}}$. Figure 4.2 illustrates some example driving scenes from CARLA and YouTube videos and their corresponding pseudo-waypoints estimated by deep visual odometry.

4.1.3 Model Pre-Training and Fine-Tuning

We pre-train the waypoint network π_θ from scratch over the large and diverse pseudo-labeled dataset $\hat{\mathcal{D}}_{\text{SemiD}}$. The pre-trained policy can then be further fine-tuned over the

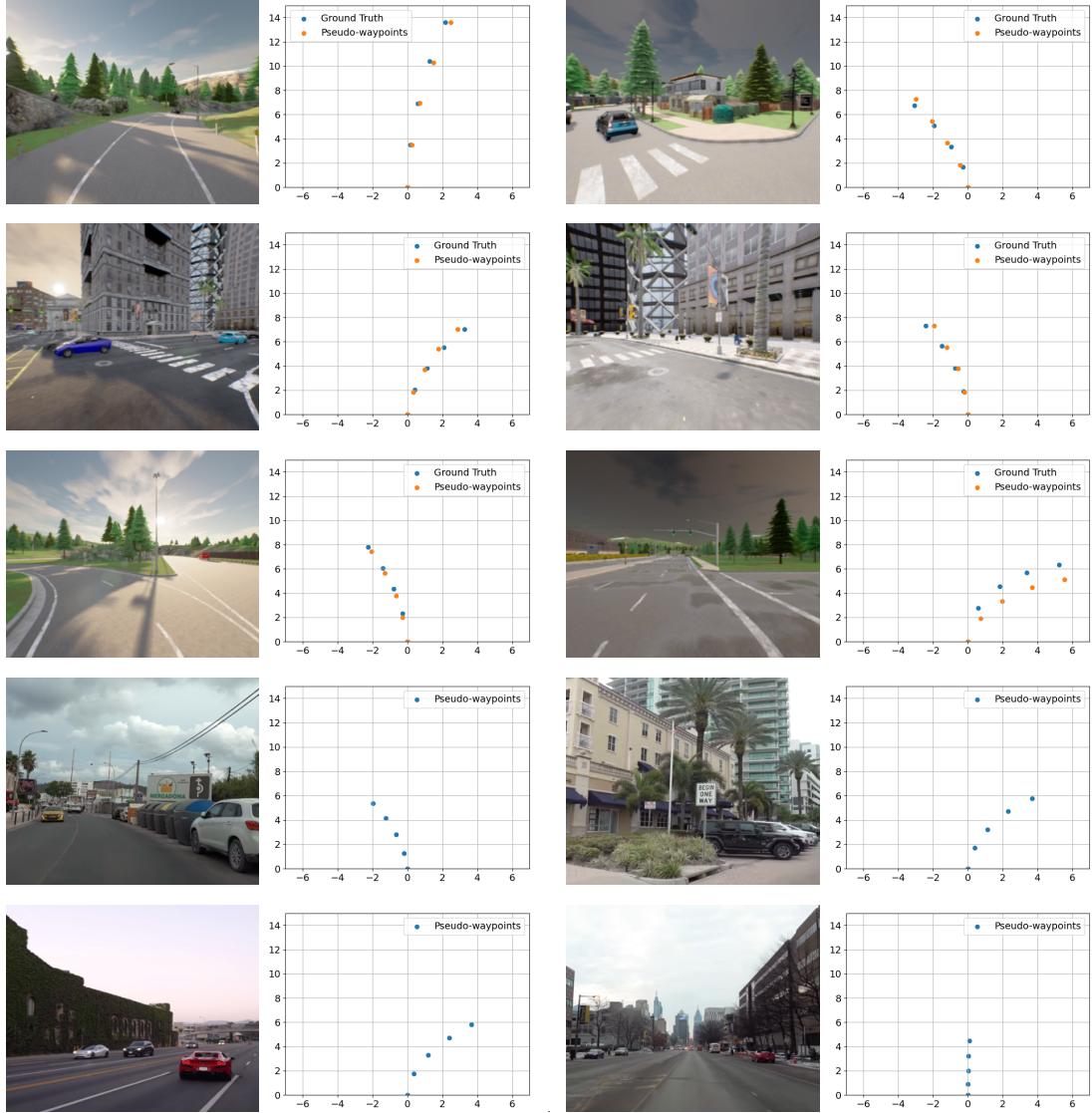


Figure 4.2: The first three rows shows examples of driving scenes from the CARLA simulator and its corresponding ground-truth waypoints and the pseudo-waypoints estimated by the DeepVO method. The latter two rows show the estimated pseudo-waypoints of some images extracted from YouTube driving videos.

small original dataset \mathcal{D} . Thus, we leverage these noisy but acceptable information from the pseudo-labeled dataset $\hat{\mathcal{D}}_{\text{SemiID}}$ to help the network gain overall knowledge about the driving, and then we provide the network with the accurate information through our small original dataset \mathcal{D} to enable the network to learn the mapping from the ground-truth information.

4.2 SelfD: Self-Learning Large-Scale Driving Policies

In this section, we discuss the semi-supervised approach ‘‘SelfD’’ proposed by Zhang et al. [68]. We follow the three main steps. We use a monocular image-based waypoint network AIM (Section 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 4.2.5). Finally, we re-train the model over the larger dataset (Section 4.2.6).

4.2.1 Conditional Imitation Learning from Observations

To make use of the unlabeled driving scenes containing diverse navigational experiences, the authors propose ‘‘Conditional Imitation Learning from Observations’’ (CILfO) [68] framework. The framework suggests a way to generate pseudo-labels for the unlabeled driving scenes. But, the network architecture used by the authors is similar to that of LBC [16] (uses discrete navigational command as the conditional command). In the work of Prakash et. al. [47], the authors have shown that, their proposed image-based architecture, AIM (uses sparse target location as the conditional command), performs significantly better than the LBC architecture. Therefore, in this work, we explore the CILfO framework in our autonomous driving setting which uses AIM in its core (Section 3.3). Therefore, we consider sparse target 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 target location \hat{c} from the input image to construct a dataset

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

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

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 pseudo-labels for the unlabeled data. This assumption is reasonable considering

that there are several publicly available driving datasets with included action labels [26, 57]. In this work, we use the small subset of a labeled dataset \mathcal{D} collected by running an expert policy rollout in CARLA [23] as discussed in Section 3.1.

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 steps summarized below:

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

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 an MLP to transform those waypoints in the BEV space. In our work, instead of the 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 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 score of the 1-meter squared bounding boxes centered around 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 3.5 and $\mathcal{L}_{\text{quality}}$ is a binary cross-entropy loss.

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 π_θ in a semi-supervised training process. Contrary to our proposed method in Section 4.1, the authors do not consider visual odometry techniques [56] to recover the expert’s actions, as they found these results 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 π_θ 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 provide additional supervision, i.e., a conditional agent that better reasons what it might need to do, for instance, if it had to turn left instead of right at an intersection (Figure 4.3). 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 $(\hat{\mathbf{W}}, \hat{o}) = \pi_\theta(I, \hat{c})$.
- discard noisy trajectories with some thresholding on the quality estimate \hat{o} and take the rest to form the pseudo-labeled dataset $\hat{\mathcal{D}}_{\text{SelfD}}$.

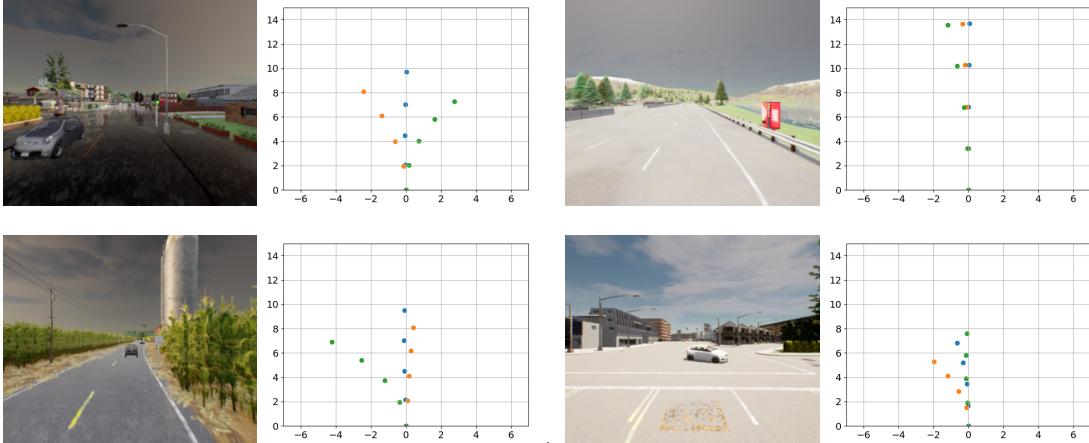


Figure 4.3: **“What If” Hypothetical Pseudo-Labeling.** We generate multiple plausible future trajectories in BEV for each unlabeled frame by uniformly sampling the target point location. Here we illustrate what if pseudo-labeling for four driving scenarios from our CARLA dataset.

4.2.6 Model Pre-Training and Fine-Tuning

As a final training step, we re-train the waypoint network π_θ 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} .

4.3 OVRL: Offline Visual Representation Learning

In this section, we discuss OVRL[61], a two-stage learning approach proposed by Yadav et. al. and incorporate it into our autonomous driving framework. As an overview, this learning approach includes an encoder pre-training step using DINO, followed by downstream policy learning via behavior cloning.

4.3.1 Self-supervised Pre-Training

As the first step, a visual encoder is pre-trained using DINO [10], a simple but effective self-supervised learning algorithm. DINO uses knowledge distillation as a mechanism for self-training, where the student network g_{θ_s} is trained to match the output of the teacher network g_{θ_t} . As illustrated in Figure 4.4, it takes an input image x and using the multi-crop strategy introduced in [11], 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:

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^K \exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

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_{θ_t} , the student network learns to match the distribution of the teacher network by minimizing the cross-entropy loss:

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

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

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: $c \leftarrow mc + (1 - m)\frac{1}{B} \sum_{i=0}^B g_{\theta_t}(x_i)$, where $m > 0$ is a momentum parameter and B is the batch size. Sharpening is achieved by setting a 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 sharpening has the opposite effect. Applying both of these operations balances their effects which is sufficient to avoid collapse in presence of a momentum teacher.

4.3.2 Implementation Details

We use the same ResNet34 architecture as discussed in 3.3. We pre-train the model on the CARLA driving scenes without the labels. We train the model with the AdamW

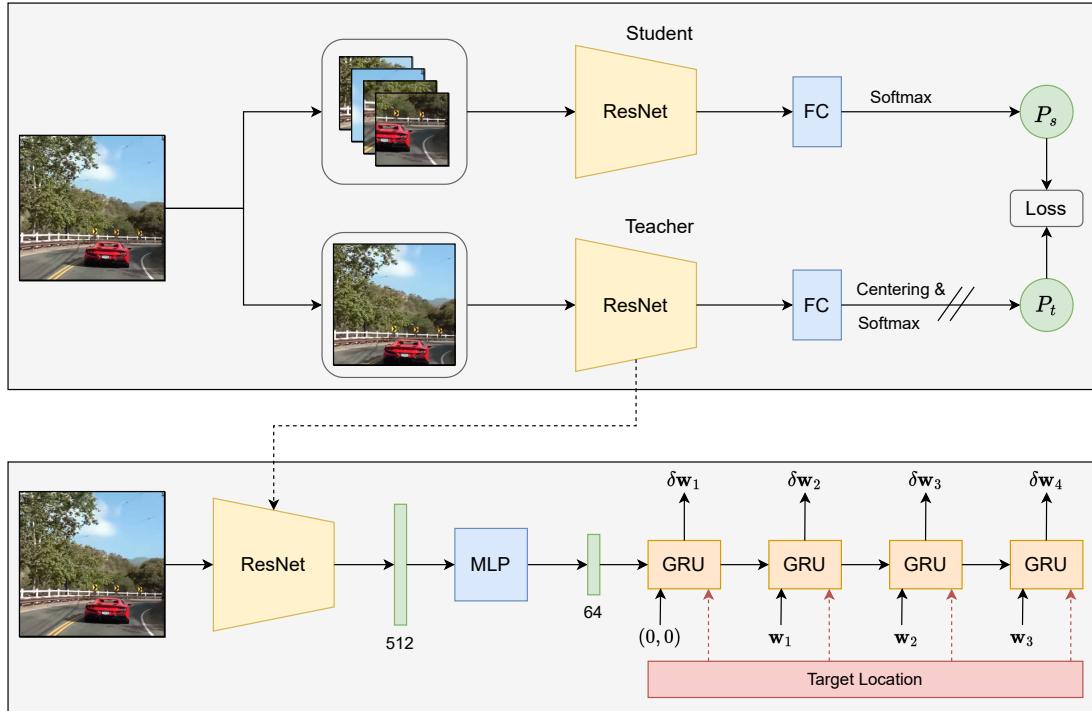


Figure 4.4: Overview of OVRL, consisting of two steps: 1) offline pre-training of the visual representations using large-scale dataset driving scenes using DINO 2) downstream finetuning of the visuomotor representations with imitation learning.

optimizer [40] and a batch size of 16 per GPU, distributed over 8 GPUs. The learning rate is linearly increased during the first 10 epochs to its base value determined with the following linear scaling rule [27]: $\text{lr} = 0.0005 \times \text{batchsize}/256$. After this warmup, we decay the learning rate with a cosine schedule [41]. Also, the weight decay 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 [11].

4.3.3 Downstream Learning

After pre-training with DINO, the whole projection head is discarded. Now we plug this pre-trained ResNet34 architecture in our waypoint prediction network (Section 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 enabling the network to quickly map the input scenes to the waypoints leveraging the better internal representation.

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.

5.1 Task

We consider the task of driving along a set of predefined routes in different environments, e.g. highways, urban areas, and residential districts. The routes are defined by a sequence of sparse target locations in GPS coordinates provided by the navigational planner together with the corresponding discrete navigational commands (e.g. follow the lane, turn left/right, change lane). In our approaches, we only use the sparse target locations along with the monocular camera images for driving. Each route consists of several scenarios and the agent is initialized at predefined positions. The ability of the agent to handle different kinds of adversarial situations (e.g. obstacle avoidance, running red lights, unprotected turns at intersections, and pedestrians emerging from occluded regions to cross the road at random locations) evaluates its performance. The agent is required to complete the route within a specified time limit while following the traffic rules and coping with the high-density 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 global planner

Along the routes, there are difficult scenarios specified which the agent needs to resolve in order to safely arrive at his destination. These are as follows:

- A pedestrian, hidden behind a static object, suddenly starts running across the street.
- The vehicle in front suddenly performs an emergency brake.
- A slow vehicle gets spawned in front of the ego-vehicle agent.
- While taking a turn, a cyclist suddenly drives across the street.
- Crossing an unsignalized intersection.
- A static object is blocking the street.
- The ego-vehicle agent must perform an unprotected right turn at an intersection.
- Crossing traffic is running a red light at an intersection with oncoming traffic
- The ego-vehicle agent must perform a left turn at an intersection with crossing traffic

5.2 Implementation Details

5.2.1 Data Cleansing

Our expert driving demonstrations contain many failure cases. For example, the ego-vehicle crashes into another vehicle or another vehicle comes in front 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 the frames in that route demonstration. Also, at 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 at full speed in such a situation as there is no car in front of it. These failure cases introduce spurious correlations that lead to causal confusion problems [22]. Therefore we remove these demonstrations which lead to improved driving performance.

5.2.2 Stratified Sampling

On top of cleaning the data, we also group the data according to different driving modalities. For example, depending on the waypoints corresponding to a driving frame, we categorize the sample as one of the four categories — ‘at turns’, ‘straight driving’, ‘at halt’ or ‘accelerating’. Next, we implement the stratified sampling technique in our data loader, which not only uses good samples but also generates a training batch (we use batch size of 64 in all our experiments) consisting of an equal number of examples from each modality. 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 driving performance.

5.2.3 Image Augmentation

Prior works [34, 63, 64, 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, 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 leads to improved robustness of the model.

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 hold out Town05 for the evaluation and use rest of the towns for training. For generating the training data we use an expert policy with the privileged information from the simulation. We rollout the expert policy and store the data at 2 frames per second. We select Town05 for evaluation due to its large diversity in drivable regions (e.g. multi-lane and single-lane roads, bridges and underpasses, highways and exits) compared to other CARLA towns. Following [47], we consider Town05-Long evaluation setting: 10 long routes of 1000-2000m comprising 10 intersections each. Each route consists of a high-density dynamic agents and adversarial scenarios which gets spawned at predefined positions along the route. As we only want to focus on handling the adversarial scenarios and dynamic agents, we can decouple this aspect from generalization across different weather conditions. Thus we only evaluate on the ClearNoon weather.

We also use a set of first-view driving videos from YouTube collected by the Zhang et. al. [69]. We use 52 videos with a total length of over 50 hours of driving demonstration. As shown in Figure 3.1, these videos cover different driving scenes with various weather conditions (e.g. sunny, rainy, snowy, etc.) and regions (e.g. rural and urban areas). We sample two frames every second, resulting to a dataset of 0.36 million frames. We use all YouTube driving data for pre-training the models in all learning approaches.

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 consider different aspects of driving. Even though all routes have the same type of metrics, their respective values are calculated separately. These metrics are as follows:

5.4.1 Route Completion (RC)

Route Completion is the percentage of the route completed by an agent. If an agent drives off-road, that corresponding percentage of the route will not be considered in 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 the ego-vehicle deviates more than 30 meters from the assigned route.
- Agent blocked: If the ego-vehicle doesn't take any actions for 180 seconds in the simulation.
- Route timeout: If the simulation of the route takes too long to finish.

5.4.2 Infraction Score (IS)

Infraction Score is the metric that conveys how safe the driving is. The agent starts with an ideal base score of 1.0 and for every infraction, the score gets multiplied by the corresponding penalty coefficient. 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 implies that subsequent infractions will have a lower impact due to the multiplicative nature of the score.

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

Table 5.1: Results on the Town05 long evaluation routes for a fully supervised sensorimotor agent (AIM) trained with various adhoc techniques

5.4.3 Driving Score (DS)

Driving Score is the main performance metric, serving as the product between the route completion and the infraction 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, driving score is a normalized metric with the best possible score being 100 and the worst possible score being 0 and it is not equivalent to multiplying the averaged route completion with the averaged infraction score. For evaluation, we run our model on the Town05 long evaluation routes with 3 different seeds and report the mean and standard deviation of the 3 scores averaged across the routes.

5.5 Results

5.5.1 Performance of Supervised Training

In our first experiment, we examine to what extent we can improve the current image-based AIM architecture [47] in CARLA evaluation setting involving complex multi-lane intersections, heavy infraction penalties, and adversarial scenarios. In Table 5.1, we report the evaluation results of supervised training with all the 7 CARLA training towns (Section 5.3) over 3 random seeds. We observe that a smaller learning rate improves driving performance. Next, we train the model with the filtered data and train with stratified sampling enabled data loader. 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 performance by a large margin. We note that each of these techniques improve the driving independently. Thus, combining all these techniques, we gain significant improvement in driving performance (from 13.3% to 32.2%) cumulating their individual contributions. Moreover, we notice that by applying these techniques route completion improved from 60% to 92% which suggests that these techniques help tackling the inertia problem.

5.5.2 Performance 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 $\mathcal{D}_{\text{small}}$ with ground-truth labels. We consider driving scenes from the 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 datasets we have the unlabeled dataset $\mathcal{U}_{\text{CARLA} + \text{YouTube}}$. 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 5.2.

Baselines: The first two rows of Table 5.2 shows evaluation scores of the AIM baselines trained supervisedly on Town 1 and 2. We report evaluation on two models with the 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. Here in both cases, we get roughly 50% in route completion and 7% in driving score, which suggests that the baseline models could not generalize well for the evaluation routes.

SelfD: Table 5.2 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 $\mathcal{U}_{\text{CARLA} + \text{YouTube}}$. But, we see no noticeable improvement and even reduction in performance in the evaluations compared to supervised training on Town1&2. SelfD training without “What If” augmentation on $\mathcal{U}_{\text{CARLA}}$ dataset shows a 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 [68], we observe that SelfD gains no generalization with “What If” augmentation. Figure 5.1 demonstrates some of the cases where SelfD fails to produce appropriate pseudo-waypoints, but SemiD produces the correct ones.

SemiD: We train our DeepVO architecture with Town1&2 data containing driving scenes and its corresponding ego-vehicle positions and gather pseudo-waypoints and pseudo-target-point for all the unlabeled dataset. We report results of SemiD for

Datasets	Method	DS (\uparrow)	RC (\uparrow)	IS (\uparrow)
Labeled Dataset: $\mathcal{D}_{\text{small}}$	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: $\mathcal{U}_{\text{CARLA}}$ Labeled Dataset: $\mathcal{D}_{\text{small}}$	SelfD (w/o What If, w/o confidence)	6.07 ± 0.41	58.36 ± 3.08	0.17 ± 0.02
	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: $\mathcal{U}_{\text{CARLA + YouTube}}$ Labeled Dataset: $\mathcal{D}_{\text{small}}$	SelfD (w/o What If, w/o confidence)	6.66 ± 1.98	51.44 ± 7.23	0.23 ± 0.03
	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: $\mathcal{U}_{\text{YouTube}}$ Labeled Dataset: $\mathcal{D}_{\text{small}}$	SemiD (transductive training)	1.96 ± 0.69	83.36 ± 4.78	0.02 ± 0.01
	SemiD (finetuned)	7.35 ± 0.63	89.78 ± 8.30	0.08 ± 0.01
	OVRL + SemiD (transductive training)	2.06 ± 0.49	85.78 ± 4.31	0.02 ± 0.01
	OVRL + SemiD (finetuned)	7.29 ± 0.91	91.62 ± 4.78	0.1 ± 0.02

Table 5.2: Results on the Town05 long evaluation routes for sensorimotor agent (AIM) trained with various semi-supervised and self-supervised learning approaches

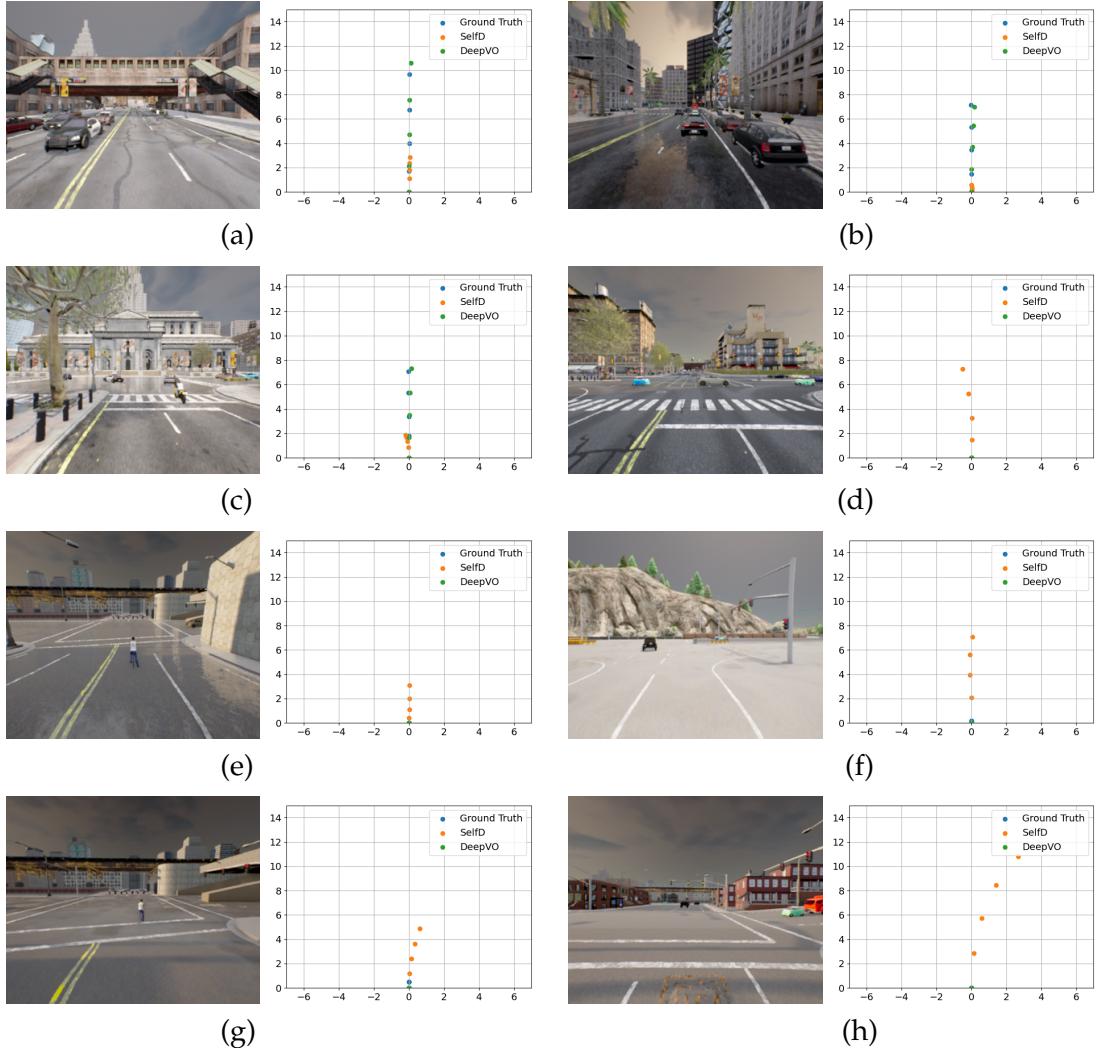


Figure 5.1: Some failure cases where SelfD could not recover the desired pseudo-waypoints but correctly recovered by DeepVO. Examples (a), (b), (c), show some situations where the ego-vehicle should move forward but SelfD predicts waypoints to slow down the ego-vehicle. Examples (d), (e) show situations where ego-vehicle should not move because of the cyclist in front 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.

three unlabeled dataset $\mathcal{U}_{\text{CARLA}}$, $\mathcal{U}_{\text{YouTube}}$ and $\mathcal{U}_{\text{CARLA} + \text{YouTube}}$ in Table 5.2. Contrary to the claim by the authors of SelfD [68], we observe this method improve driving performance. With only the transductive learning stage (i.e. training with pseudo-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. This indicates that utilizing this method, the model can generalize better in the evaluation routes compared to the baselines without requiring anymore ground truth data. 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 performance in route completion as the fully supervisedly trained model on all CARLA training towns (see Table 5.1).

OVRL: We also show performance of the self-supervised learning approach OVRL in Table 5.2. We pre-train the ResNet34 backbone on the $\mathcal{U}_{\text{CARLA} + \text{YouTube}}$ unlabeled dataset self-supervisedly and fine-tuned on Town1&2. Here also we gain improvement in route completion from 51% to 83%. We also combine OVRL and SemiD, by using the pre-trained ResNet34 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 average route completion achieved by the 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 twofold. 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 pseudo-waypoints are noisy especially when the ego-vehicle stands still and therefore it is hard for the model to map appropriate behavior in all infraction scenarios and for OVRL it happens because the traffic light signal in european towns is significantly different from the evaluation town which has the highway traffic lights. Beyond these reasons, infractions due to invisibility (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 has gone past it) is one of the biggest reason why infraction score is affected even in supervised training.

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 Town05 long evaluation routes by achieving 95% in route completion. We also observed that OVRL can learn generalizable representations and bring substantial improvement in driving performance with 97.5% in route completion when combined with SemiD. Thus, we showed that we can significantly improve performance 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 architecture 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 infractions frequently. Thus, a future direction could be to investigate this issue. Also, our model development was restricted to large-scale training with a 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: (1995). Ed. by Koichi Furukawa, Donald Michie, and Stephen H. Muggleton, pp. 103–129.
- [2] Nicolas Ballas et al. "Delving Deeper into Convolutional Networks for Learning Video Representations". In: (2016). Ed. by Yoshua Bengio and Yann LeCun. URL: <http://arxiv.org/abs/1511.06432>.
- [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.R0].
- [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] Tobias Betz, Hidehito Fujiishi, and Taisuke Kobayashi. "Behavioral Cloning from Observation with Bi-directional Dynamics Model". In: (2021), pp. 184–189. doi: 10.1109/IEEECONF49454.2021.9382751. URL: <https://doi.org/10.1109/IEEECONF49454.2021.9382751>.
- [7] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: (Apr. 2016). arXiv: 1604.07316 [cs.CV].
- [8] Andreas Bühler et al. "Driving Through Ghosts: Behavioral Cloning with False Positives". In: (Aug. 2020). arXiv: 2008.12969 [cs.CV].
- [9] Benjamin Caine et al. "Pseudo-labeling for Scalable 3D Object Detection". In: *CoRR* abs/2103.02093 (2021). arXiv: 2103.02093. URL: <https://arxiv.org/abs/2103.02093>.
- [10] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: (2021), pp. 9630–9640. doi: 10.1109/ICCV48922.2021.00951. URL: <https://doi.org/10.1109/ICCV48922.2021.00951>.
- [11] Mathilde Caron et al. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: (2020). Ed. by Hugo Larochelle et al. URL: <https://proceedings.neurips.cc/paper/2020/hash/70feb62b69f16e0238f741fab228fec2-Abstract.html>.
- [12] Sergio Casas, Abbas Sadat, and Raquel Urtasun. "MP3: A Unified Model To Map, Perceive, Predict and Plan". In: (2021), pp. 14403–14412. doi: 10.1109/

Bibliography

- CVPR46437 . 2021 . 01417. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Casas_MP3_A_Unified_Model_To_Map_Perceive_Predict_and_Plan_CVPR_2021_paper.html.
- [13] Matthew Chang, Arjun Gupta, and Saurabh Gupta. "Semantic Visual Navigation by Watching YouTube Videos". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/2cd4e8a2ce081c3d7c32c3cde4312ef7-Abstract.html>.
- [14] Chenyi Chen et al. "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 2722–2730. doi: [10.1109/ICCV.2015.312](https://doi.org/10.1109/ICCV.2015.312). URL: <https://doi.org/10.1109/ICCV.2015.312>.
- [15] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. "Learning to drive from a world on rails". In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 15570–15579. doi: [10.1109/ICCV48922.2021.01530](https://doi.org/10.1109/ICCV48922.2021.01530). URL: <https://doi.org/10.1109/ICCV48922.2021.01530>.
- [16] Dian Chen et al. "Learning by Cheating". In: *Proceedings of Machine Learning Research 100* (2019). Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, pp. 66–75. URL: <http://proceedings.mlr.press/v100/chen20a.html>.
- [17] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of Machine Learning Research 119* (2020), pp. 1597–1607. URL: <http://proceedings.mlr.press/v119/chen20j.html>.
- [18] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning". In: (2021), pp. 15750–15758. doi: [10.1109/CVPR46437.2021.01549](https://doi.org/10.1109/CVPR46437.2021.01549). URL: https://openaccess.thecvf.com/content/CVPR2021/html/Chen_Exploring_Simple_Siamese_Representation_Learning_CVPR_2021_paper.html.
- [19] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (June 2014). arXiv: [1406.1078](https://arxiv.org/abs/1406.1078) [cs.CL].
- [20] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR abs/1412.3555* (2014). arXiv: [1412.3555](https://arxiv.org/abs/1412.3555). URL: [http://arxiv.org/abs/1412.3555](https://arxiv.org/abs/1412.3555).
- [21] Felipe Codevilla et al. "End-to-End Driving Via Conditional Imitation Learning". In: (2018), pp. 1–9. doi: [10.1109/ICRA.2018.8460487](https://doi.org/10.1109/ICRA.2018.8460487). URL: <https://doi.org/10.1109/ICRA.2018.8460487>.
- [22] Felipe Codevilla et al. "Exploring the Limitations of Behavior Cloning for Autonomous Driving". In: (2019), pp. 9328–9337. doi: [10.1109/ICCV.2019.00942](https://doi.org/10.1109/ICCV.2019.00942). URL: <https://doi.org/10.1109/ICCV.2019.00942>.

- [23] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: Proceedings of Machine Learning Research 78 (2017), pp. 1–16. URL: <http://proceedings.mlr.press/v78/dosovitskiy17a.html>.
- [24] Alexey Dosovitskiy et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: (2015), pp. 2758–2766. doi: [10.1109/ICCV.2015.316](https://doi.org/10.1109/ICCV.2015.316). URL: <https://doi.org/10.1109/ICCV.2015.316>.
- [25] Angelos Filos et al. "Can Autonomous Vehicles Identify, Recover From, and Adapt to Distribution Shifts?" In: Proceedings of Machine Learning Research 119 (2020), pp. 3145–3153. URL: <http://proceedings.mlr.press/v119/filos20a.html>.
- [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: CoRR abs/1706.02677 (2017). arXiv: [1706.02677](https://arxiv.org/abs/1706.02677). URL: [http://arxiv.org/abs/1706.02677](https://arxiv.org/abs/1706.02677).
- [28] Jean-Bastien Grill et al. "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning". In: (2020). Ed. by Hugo Larochelle et al. URL: <https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html>.
- [29] Saurabh Gupta et al. "Cognitive Mapping and Planning for Visual Navigation". In: *Int. J. Comput. Vis.* 128.5 (2020), pp. 1311–1330. doi: [10.1007/s11263-019-01236-7](https://doi.org/10.1007/s11263-019-01236-7). URL: <https://doi.org/10.1007/s11263-019-01236-7>.
- [30] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. "Causal Confusion in Imitation Learning". In: (May 2019). arXiv: [1905.11979](https://arxiv.org/abs/1905.11979) [cs.LG].
- [31] Jeffrey Hawke et al. "Urban Driving with Conditional Imitation Learning". In: (Nov. 2019). arXiv: [1912.00177](https://arxiv.org/abs/1912.00177) [cs.CV].
- [32] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning". In: (2020), pp. 9726–9735. doi: [10.1109/CVPR42600.2020.00975](https://doi.org/10.1109/CVPR42600.2020.00975). URL: <https://doi.org/10.1109/CVPR42600.2020.00975>.
- [33] Joel Janai et al. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. Vol. 12. 1-3. 2020, pp. 1–308. doi: [10.1561/0600000079](https://doi.org/10.1561/0600000079). URL: <https://doi.org/10.1561/0600000079>.
- [34] Michael Laskin et al. "Reinforcement Learning with Augmented Data". In: (2020). Ed. by Hugo Larochelle et al. URL: <https://proceedings.neurips.cc/paper/2020/hash/e615c82aba461681ade82da2da38004a-Abstract.html>.
- [35] Yann LeCun et al. "Off-Road Obstacle Avoidance through End-to-End Learning". In: *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*. 2005, pp. 739–746. URL: <https://proceedings.neurips.cc/paper/2005/hash/fdf1bc5669e8ff5ba45d02fded729feb-Abstract.html>.
- [36] Yann LeCun et al. "Off-Road Obstacle Avoidance through End-to-End Learning". In: *Advances in Neural Information Processing Systems 18 [Neural Information*

- Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]. 2005, pp. 739–746. URL: <https://proceedings.neurips.cc/paper/2005/hash/fdf1bc5669e8ff5ba45d02fded729feb-Abstract.html>.*
- [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: (2019). Ed. by Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson. doi: [10.15607/RSS.2019.XV.005](https://doi.org/10.15607/RSS.2019.XV.005). URL: <https://doi.org/10.15607/RSS.2019.XV.005>.
- [39] Xiaodan Liang et al. “CIRL: Controllable Imitative Reinforcement Learning for Vision-Based Self-driving”. In: *Lecture Notes in Computer Science* 11211 (2018). Ed. by Vittorio Ferrari et al., pp. 604–620. doi: [10.1007/978-3-030-01234-2_36](https://doi.org/10.1007/978-3-030-01234-2_36). URL: https://doi.org/10.1007/978-3-030-01234-2_36.
- [40] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: (2019). URL: <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [41] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: (2017). URL: <https://openreview.net/forum?id=Skq89Scxx>.
- [42] Lina Mezghani et al. “Memory-Augmented Reinforcement Learning for Image-Goal Navigation”. In: *CoRR* abs/2101.05181 (2021). arXiv: [2101.05181](https://arxiv.org/abs/2101.05181). URL: <https://arxiv.org/abs/2101.05181>.
- [43] Matthias Müller et al. “Driving Policy Transfer via Modularity and Abstraction”. In: (Apr. 2018). arXiv: [1804.09364 \[cs.R0\]](https://arxiv.org/abs/1804.09364).
- [44] 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](https://doi.org/10.1109/CVPR42600.2020.01131). URL: <https://doi.ieee.org/10.1109/CVPR42600.2020.01131>.
- [45] Takayuki Osa et al. “An Algorithmic Perspective on Imitation Learning”. In: *Found. Trends Robotics* 7.1-2 (2018), pp. 1–179. doi: [10.1561/2300000053](https://doi.org/10.1561/2300000053). URL: <https://doi.org/10.1561/2300000053>.
- [46] Dean Pomerleau. “ALVINN: An Autonomous Land Vehicle in a Neural Network”. In: *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*. Ed. by David S. Touretzky. Morgan Kaufmann, 1988, pp. 305–313. URL: <http://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network>.
- [47] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. “Multi-Modal Fusion Transformer for End-to-End Autonomous Driving”. In: (2021), pp. 7077–7087. doi: [10.1109/CVPR46437.2021.00700](https://doi.org/10.1109/CVPR46437.2021.00700). URL: https://openaccess.thecvf.com/content/CVPR2021/html/Prakash_Multi-Modal_Fusion_Transformer_for_End-to-End_Autonomous_Driving_CVPR_2021_paper.html.
- [48] Aditya Prakash et al. “Exploring Data Aggregation in Policy Learning for Vision-Based Urban Autonomous Driving”. In: *2020 IEEE/CVF Conference on*

- Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* Computer Vision Foundation / IEEE, 2020, pp. 11760–11770. doi: 10.1109/CVPR42600.2020.01178. url: https://openaccess.thecvf.com/content/_CVPR_2020/html/Prakash_Exploring_Data_Aggregation_in_Policy_Learning_for_Vision-Based_Urban_Autonomous_CVPR_2020_paper.html.
- [49] Mamshad Nayeem Rizve et al. “In Defense of Pseudo-Labeling: An Uncertainty-Aware Pseudo-label Selection Framework for Semi-Supervised Learning”. In: (2021). url: <https://openreview.net/forum?id=-ODN6SbiUU>.
- [50] Axel Sauer, Nikolay Savinov, and Andreas Geiger. “Conditional Affordance Learning for Driving in Urban Environments”. In: *Proceedings of Machine Learning Research* 87 (2018), pp. 237–252. url: <http://proceedings.mlr.press/v87/sauer18a.html>.
- [51] Ardi Tampuu et al. “A Survey of End-to-End Driving: Architectures and Training Methods”. In: *IEEE Trans. Neural Networks Learn. Syst.* 33.4 (2022), pp. 1364–1384. doi: 10.1109/TNNLS.2020.3043505. url: <https://doi.org/10.1109/TNNLS.2020.3043505>.
- [52] Faraz Torabi, Garrett Warnell, and Peter Stone. “Recent Advances in Imitation Learning from Observation”. In: (2019). Ed. by Sarit Kraus, pp. 6325–6331. doi: 10.24963/ijcai.2019/882. url: <https://doi.org/10.24963/ijcai.2019/882>.
- [53] Marin Toromanoff, Émilie Wirbel, and Fabien Moutarde. “End-to-End Model-Free Reinforcement Learning for Urban Driving Using Implicit Affordances”. In: (2020), pp. 7151–7160. doi: 10.1109/CVPR42600.2020.00718. url: https://openaccess.thecvf.com/content/_CVPR_2020/html/Toromanoff_End-to-End_Model-Free_Reinforcement_Learning_for_Urban_Driving_Using_Implicit_Affordances_CVPR_2020_paper.html.
- [54] Dequan Wang et al. “Monocular Plan View Networks for Autonomous Driving”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019.* IEEE, 2019, pp. 2876–2883. doi: 10.1109/IROS40897.2019.8967897. url: <https://doi.org/10.1109/IROS40897.2019.8967897>.
- [55] Jingke Wang et al. “Learning hierarchical behavior and motion planning for autonomous driving”. In: (2020), pp. 2235–2242. doi: 10.1109/IROS45743.2020.9341647. url: <https://doi.org/10.1109/IROS45743.2020.9341647>.
- [56] Sen Wang et al. “DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks”. In: (2017), pp. 2043–2050. doi: 10.1109/ICRA.2017.7989236. url: <https://doi.org/10.1109/ICRA.2017.7989236>.
- [57] 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 1, NeurIPS Datasets and Benchmarks*

Bibliography

- 2021, December 2021, virtual. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. 2021. URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/4734ba6f3de83d861c3176a6273cac6d-Abstract-round2.html>.
- [58] Zhirong Wu et al. “Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination”. In: *CoRR* abs/1805.01978 (2018). arXiv: 1805.01978. URL: <http://arxiv.org/abs/1805.01978>.
- [59] Yi Xiao et al. “Action-based Representation Learning for Autonomous Driving”. In: Proceedings of Machine Learning Research 155 (2020). Ed. by Jens Kober, Fabio Ramos, and Claire J. Tomlin, pp. 232–246. URL: <https://proceedings.mlr.press/v155/xiao21a.html>.
- [60] Yi Xiao et al. “Multimodal End-to-End Autonomous Driving”. In: *IEEE Trans. Intell. Transp. Syst.* 23.1 (2022), pp. 537–547. doi: 10.1109/TITS.2020.3013234. URL: <https://doi.org/10.1109/TITS.2020.3013234>.
- [61] Karmesh Yadav et al. “Offline Visual Representation Learning for Embodied Navigation”. In: *CoRR* abs/2204.13226 (2022). doi: 10.48550/arXiv.2204.13226. arXiv: 2204.13226. URL: <https://doi.org/10.48550/arXiv.2204.13226>.
- [62] Jihan Yang et al. “ST3D: Self-Training for Unsupervised Domain Adaptation on 3D Object Detection”. In: (2021), pp. 10368–10378. doi: 10.1109/CVPR46437.2021.01023. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Yang_ST3D_Self-Training_for_Unsupervised_Domain_Adaptation_on_3D_Object_Detection_CVPR_2021_paper.html.
- [63] Denis Yarats, Ilya Kostrikov, and Rob Fergus. “Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels”. In: (2021). URL: <https://openreview.net/forum?id=GY6-6sTvGaf>.
- [64] Denis Yarats et al. “Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning”. In: (2022). URL: https://openreview.net/forum?id=_SJ-_yyes8.
- [65] Jure Zbontar et al. “Barlow Twins: Self-Supervised Learning via Redundancy Reduction”. In: Proceedings of Machine Learning Research 139 (2021). Ed. by Marina Meila and Tong Zhang, pp. 12310–12320. URL: <http://proceedings.mlr.press/v139/zbontar21a.html>.
- [66] Guangyao Zhai et al. “PoseConvGRU: A Monocular Approach for Visual Ego-motion Estimation by Learning”. In: *Pattern Recognit.* 102 (2020), p. 107187. doi: 10.1016/j.patcog.2019.107187. URL: <https://doi.org/10.1016/j.patcog.2019.107187>.
- [67] Jimuyang Zhang and Eshed Ohn-Bar. “Learning by Watching”. In: (June 2021). arXiv: 2106.05966 [cs.CV].
- [68] Jimuyang Zhang, Ruizhao Zhu, and Eshed Ohn-Bar. “SelfD: Self-Learning Large-Scale Driving Policies From the Web”. In: (2022), pp. 17295–17305. doi: 10.1109/CVPR52688.2022.01680. URL: <https://doi.org/10.1109/CVPR52688.2022.01680>.

- [69] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. “Action-Conditioned Contrastive Policy Pretraining”. In: *CoRR* abs/2204.02393 (2022). doi: [10.48550/arXiv.2204.02393](https://doi.org/10.48550/arXiv.2204.02393). arXiv: 2204.02393. URL: <https://doi.org/10.48550/arXiv.2204.02393>.
- [70] Albert Zhao et al. “LaTeS: Latent Space Distillation for Teacher-Student Driving Policy Learning”. In: *CoRR* abs/1912.02973 (2019). arXiv: 1912.02973. URL: <http://arxiv.org/abs/1912.02973>.
- [71] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. “Does computer vision matter for action?” In: *Sci. Robotics* 4.30 (2019). doi: [10.1126/scirobotics.aaw6661](https://doi.org/10.1126/scirobotics.aaw6661). URL: <https://doi.org/10.1126/scirobotics.aaw6661>.