Efficient 3D Vision for Autonomous Driving

Philip Jacobson

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-164
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-164.html

August 9, 2024

Efficient 3D Vision for Autonomous Driving

by

Philip Long Jacobson

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Summer 2024

Efficient 3D Vision for Autonomous Driving

Abstract

Efficient 3D Vision for Autonomous Driving

by

Philip Jacobson

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ming C. Wu, Chair

Self-driving vehicles have long been envisioned as a massive leap forward in transportation technology. Although several efforts to developing fully autonomous vehicles are currently being undertaken in both industry and academia, so far none have achieved the promise of full self-driving. Of the challenges in building the autonomous software for self-driving cars, one of the most prominent is perception, or the ability for the vehicle to sense the world around it. To meet the requirements for practical deployment onto autonomous vehicles, perception systems must meet four key metrics of efficiency: accuracy, low-latency, reasonable compute hardware, and training data efficiency.

In this dissertation, we will introduce novel approaches to AV perception while aiming to address the four metrics for efficiency. We introduce four major new perception schemes during the course of this dissertation.

In Chapter 2, we consider a combined hardware/algorithms approach to perception to achieve accelerated training speeds on limited compute hardware. We introduce system based on the principles of delayed-feedback reservoir computing implemented using an optoelectronic delay system. To tailor this approach to computer vision tasks, we combine it with a high-speed digital preprocessing through untrained convolutional layers to generate randomized feature maps that are then circulated through our reservoir. We experimentally validate our approach on the classic MNIST handwritten digit recognition task, and achiever performance on-par with a digitally-trained convolutional neural network, while achieving a training-time speed-up of up to 10×.

In Chapter 3, we consider 3D object detection in autonomous driving settings, and specifically consider the problem of efficient LiDAR-camera fusion. We introduce a novel sensor fusion approach, dubbed Center Feature Fusion, which operates through fusing camera and LiDAR deep features in the bird's-eye-view space. To enable low-latency fusion, we propose a sparse feature fusion, we projects only a set of identified key camera features to bird's-eye-view. As a result, we are able to achieve performance on-par with competing sensor fusion approaches, while reducing runtime latency by several times.

In Chapter 4, we consider the problem of 3D object detection from the data efficiency

angle, aiming to reduce the labeled data requirement needed to train the computer vision models necessary for autonomous vehicles. In this chapter, we introduce doubly-robust self-training, a novel generalized approach to semi-supervised learning. We conduct both theoretical analysis to demonstrate its superiority over the standard self-training approaches regardless of teacher model quality, and experimental analysis on both image classification and object detection. For both vision tasks, we achieve performance superior to the self-training baseline with no extra computational costs.

In Chapter 5, we continue exploring semi-supervised 3D object detection through leveraging the motion forecasting component of the autonomy stack to improve perception models. We introduce our novel algorithm , TrajSSL, which uses a pre-trained prediction model to generate a set of synthetic labels to enhance the training of a student detector model. The generate synthetic labels are used to establish temporal consistency, and thus filter out low-quality pseudo-labels during training, while simultaneously correcting for missing pseudo-labels. TrajSSL outperforms the state-of-the-art for semi-supervised 3D object detection across a wide variety of scenarios.

To my family

Contents

List of Tables

List of Figures

Acknowledgments

I'm lucky to have been surrounded by so many wonderful mentors and peers during my PhD. The work in this dissertation was the culmination of the efforts of many who have guided and supported me along the way.

I would first like to thank my advisor, Professor Ming C. Wu. For the entirety of my PhD studies, he has guided and supported me in my research while never failing to believe in my capabilities as a researcher.

I would to thank Professor Masayoshi Tomizuka and Dr. Wei Zhan, whose group I worked closely with for the last several years of my PhD. The Mechanical Systems Control Lab has been a second home for me, and I'm grateful for my experience there. I would like to thank Professor Avideh Zakhor for serving on both my Quals and thesis committee, and Professor Kris Pister for serving on my Quals committee.

I would like to thank all of my wonderful lab mates in my group who have helped me grow as a researcher: Dr. Guan-Lin Su, who took me on when I was a starry-eyed first year and showed me the ropes, helped me get a running start as a PhD student. Dr. Mizuki Shirao, who during his time as a visiting scholar, imparted upon me an immense amount of expertise in systems-level optoelectronics that was critical to my work. Kerry Yu, who I worked closely with to program our FPGA for our experiments. Dr. Jean-Etienne Tremblay, Dr. Xiaosheng Zhang, Dr. Jodi Loo, Dr. Nicolas Andrade, Daniel Klawson, Jianheng Luo, Johannes Henriksson, and all of the other lab members who I crossed paths with and learned from during my journey.

I would like to thank all of my collaborators in the MSC lab: Dr. Yiyang Zhou took me in as a new member and helped me learn the ropes in an unfamiliar field. Dr. Mingyu Ding, Chenfeng Xu, Yichen Xie, who worked closely with during my projects in the MSC lab and have helped me learn and mature as a computer vision researcher immensely. Banghua Zhu and Professor Jiantao Jiao, who I worked closely with as collaborators of the MSC lab.

I would like to thank the National Defense Science and Engineering Graduate Fellowship and all of the folks at the DoD and AFRL, who funded the final three years of my PhD studies. I would also like to thank Dr. Andreas Schmitt-Sody, who served as a wonderful mentor, both professionally and personally.

During my PhD, I was lucky to have two fruitful internships at HRL Laboratories and Interdigital Inc. I would like to thank my advisors Dr. Heiko Hoffmann and Dr. Jiahao Pang, whose advice and mentorship helped me bear fruit during my dissertation work.

I would like to thank my wonderful family: Mom, Dad, JingJing, and LingLing, even though I moved across the country to pursue my education, never faltered in supporting me personally in my journey. Through our many phone calls, I always felt you were by my side. Lastly, I would like to thank my wonderful girlfriend Katie, who has brought me immense joy and support during my PhD. I look forward to our future adventures ahead.

Chapter 1

Introduction

Autonomous robots have long captured the imagination of scientists and the public alike. With recent advancements in machine learning (ML) over the past decade, growth in automation and robotics has been tremendous, and by 2030, the market is estimated to reach up to a $260 billion dollar market cap [51]. One particular segment of the robotics industry, autonomous driving, has received significant funding and attention over the past several years. Ranging in applications from robotaxis traversing our cities to unmanned vehicles for mining and agriculture to robots capable of deployment on the battlefield, the potential of self-driving technology is staggering. The challenge of developing this technology has brought together industry and academic researchers in computer vision, signal processing, controls, motion planning, and hardware design. Nonetheless, the goal of achieving SAE-designated L5 autonomy (full driving capability with no human intervention) has yet to be realized.

The first component of the autonomy stack (and the focus of this thesis) is the perception system, i.e. the "eyes" of the autonomous vehicle (AV). For on-road self-driving cars, the perception system needs to handle a diverse set of high-level tasks, such as lane detection, localization, traffic light/sign detection, vehicle and pedestrian detection, etc. At the physical sensor level, the vehicle relies on any number of different sensor elements, commonly including cameras, LiDAR, RADAR, etc. Custom-built embedded systems house the computing hardware which process the incoming sensor data, and include both CPU and GPU components to enable real-time processing. Lastly, the software backend powering the perception system is comprised of both classical computer vision techniques, as well as modern machine learning/deep learning-enabled vision. Given the massive computing power required to train modern deep learning models, the perception models are typically trained off-board on powerful servers in data centers before being deployed on the vehicle.

Perception is perhaps the most important component in the robot autonomy stack; without highly-accurate perception, none of the downstream modules responsible for planning and decision-making can function properly. For on-road self-driving cars, the stakes are particularly high; faulty perception can lead to deadly collisions. Given the high-stakes involved in vehicle perception, research aiming to address the challenges and faults of the current state-of-the-art is critical. The following sections introduce several aspects of perception that this thesis is focused on studying.

1.1 AV Sensors

Due to the challenging nature of AV perception, modern iterations of autonomous vehicles typically rely on a host of different physical sensors. Common sensors deployed on AVs include cameras, radar, LiDAR, and ultrasonic sensors, each with its own particular strengths and use-cases. Fig 1.1 illustrates an example of a modern AV sensor suite, in this case on a Waymo robotaxi.

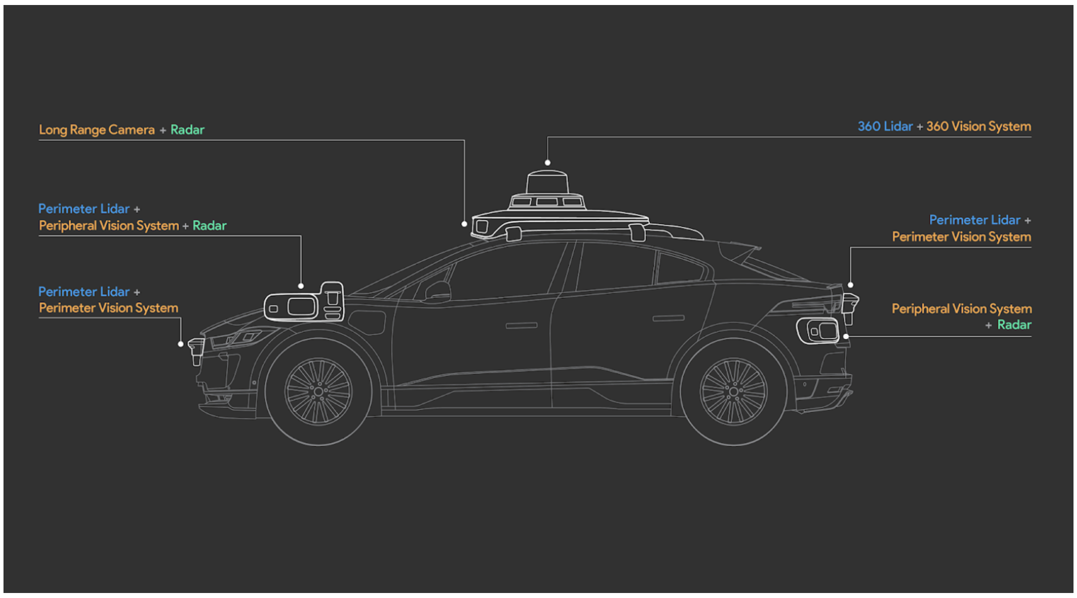

Long Range Camera + Radar
360 Lidar + 360 Vision System
Perimeter Lidar + Peripheral Vision System + Radar
Perimeter Lidar + Perimeter Vision System
Perimeter Lidar + Perimeter Vision System
Peripheral Vision System + Radar

Fig. 1.1: Sensor system on Waymo's 5^{th} generation AV. Adapted from [40]

1.1.1 Camera

Cameras are the most ubiquitous technology employed for imaging, broadly speaking, and play an important role in the AV perception pipeline. By capturing light with electronic image sensors, digital cameras are able to generate high-resolution RGB (red, green blue) images of the world. Furthermore, the maturity of the technology allows for cameras to be manufactured to be both compact and cheap, especially in contrast to competing sensor technologies.

The main drawback of cameras is the images they capture are purely 2D, lacking depth measurements to paint a full picture of a 3D scene. Several strategies exist to try and extract 3D information from pure camera images in an AV setting. Monocular depth estimation tries to directly estimate depth from a single RGB image; however the ill-posed nature of the problem makes monocular depth estimation challenging, with the best-performing approaches still far from reliable enough to form the backbone of AV perception. Stereo vision utilizes scene captures from two cameras of differing locations and geometries to ascertain depth information of objects in the scene. However, stereo vision fails for flat objects with consistent textures and no distinguishing features.

Additionally, cameras also suffer from failure in harsh lighting conditions (both extremely dark or extremely bright), or in adverse weather conditions (heavy rain/snow, fog, dust, etc).

1.1.2 LiDAR

LiDAR (short for **Li**ght **D**etection **a**nd **R**anging) is a powerful 3D imaging technique based on the principle of measuring the time-of-flight of emitted laser pulses reflecting off surfaces in the scene. The LiDAR returns are rasterized as a point cloud, a collection of $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ points (with optional point attributes such as velocity and reflectance, depending on the LiDAR measurement capabilities) which illustrate a highly-accurate 3D layout of the scene. Because LiDAR utilizes short-wavelength light in the near-infrared, LiDAR is capable of achieving the high measurement resolution needed for sensing small objects typical in driving scenarios. Furthermore, LiDAR is agnostic to lighting conditions, capable of functioning in both intense lighting and complete darkness. Because of these capabilities, LiDAR is often employed as the main workhorse in modern AV perception systems, capturing the detailed spatial information robots need to navigate the world.

Several key drawbacks have limited the proliferation of LiDAR, and stop it from being the magic bullet for 3D perception. One significant economic issue is cost: the high-powered LiDARs employed for AV applications are significantly more expensive than more mature sensing technologies, due to the large number of moving components and precise calibration required. Alternative LiDAR sensors (i.e. utilizing MEMS beamsteering [133]) leveraging cheaper manufacturing processes have been developed, but have not yet matured to the point of practicality for AVs.

In terms of perception performance, LiDAR is limited due to its inability to detect key *semantic* features of the scene, namely color and texture. For objects encountered by drivers requiring a semantic understanding, for example traffic signs, LiDAR perception is not adequate. LiDAR also suffers from a limited resolution; compared to the millions of pixels digital cameras can achieve in a single image, a scene-level LiDAR point cloud may have only hundreds of thousands of points. For small or far-away objects, this resolution may not be sufficient. Lastly, LiDARs also fail in particularly adverse weather conditions, as the laser light can scatter off particulates in the air, creating false readings.

1.1.3 Radar

Radar operates similarly to LiDAR, however instead uses radio frequency waves for sensing. Because of the maturity of the technology, radar is relatively inexpensive and is already used significantly for both AV applications and lower-level automation (i.e. driver assist systems). Because of its longer wavelength, radar is more effective at penetrating adverse weather conditions (e.g. fog), and like LiDAR is not affected by lighting conditions. However, radar has much lower resolution than LiDAR and is not effective at discerning smaller objects and detailed geometries.

1.1.4 Ultrasonic Sensing

Ultrasonic sensing works by measuring the travel time of emitted ultrasonic sound waves reflecting off surrounding objects. Ultrasonic sensors are an ideal solution for close-range object detection, and are heavily utilized in parking-assist software to measure the distance to obstructions up to a few feet away. However, ultrasonic sensors have a very limited range, and thus their utility for true L5 autonomy is limited.

1.1.5 Sensor Fusion

Due to the challenges of the autonomous driving task, the best results are achieved through utilizing many different sensor modalities for perception. To summarize the strengths and weaknesses of each sensor modality, a qualitative comparison is shown in Fig 1.2 *Sensor fusion* describes the process of combining sensor readings from multiple sources to minimize uncertainty in the perception pipeline. Due to the disparate ways of representing sensor readings from differing modalities, the best strategy for combining these is not straightforward to determine. This topic will be discussed in further detail in Chapter 3.

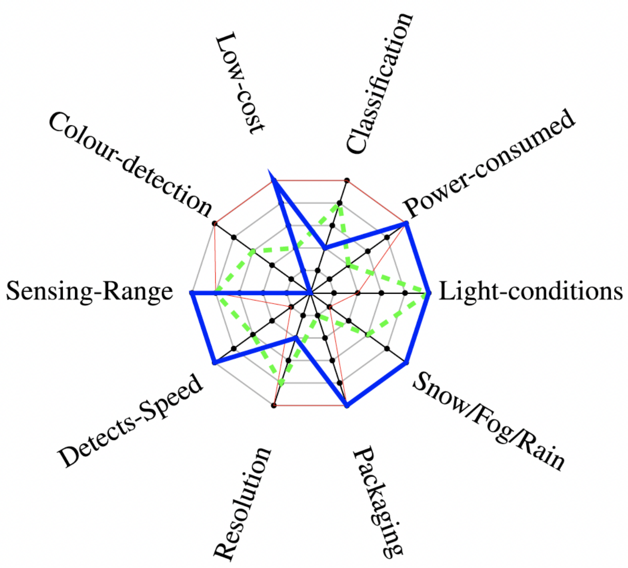


Fig. 1.2: Qualitative comparison of differing 3D sensors: camera (red), LiDAR (green), radar (blue). Adapted from [87].

1.2 Computing Hardware for Perception

The other facet of hardware enabling the perception part of the autonomy stack is the computing hardware. The current standard for ML training/deployment are typical hardware architectures used for digital computing.

1.2.1 Digital Hardware for Training

In the standard AV computing environment, the two key computing units housing the perception software are the central processing unit (CPU) and graphics processing unit

(GPU). CPUs excel at handling a general set of computing instructions at fast speeds and in sequence. In the context of 3D vision, CPUs handle the preprocessing of raw sensor data. GPUs, on the other hand, are heavily optimized for performing massively parallel computations, offering significant speedups on repetitive arithmetic calculations in comparison to CPUs. GPUs are used to handle the intense tensor multiplication arising from the deep learning models used in the perception pipeline.

Due to the nature of training modern computer vision models, model training is generally performed off-board of the AV fleet. Large computing servers with hundreds of server-side GPUs (e.g. NVIDIA A100) are used to train massive models for the perception task, allowing for both massive training acceleration and larger memory-footprint models. After training, perception models are deployed on custom-built ASICs mounted onto vehicles to be run in real-time. Power consumption and latency requirements limits the hardware that can be used on-vehicle. For example, instead of powerful server-side GPUs with large amounts of RAM, low-power edge GPUs (e.g. NVIDIA Jetson AGX Orin) are used onboard the AV fleet. To make server-trained models compatible with the resource-constrained vehicle hardware, these models first undergo a process of "distillation" to reduce the number of parameters to be more manageable with limited memory and compute.

A drawback of the two-tiered training and deployment model for AV perception is the inability for on-board training during real-time vehicle deployment. For many AV/robotics tasks, updating the perception model with new observations during deployment is highly desirable. However, the standard approach of training models server-side makes this on-the-fly-learning challenging. Cloud connectivity is often limited during deployment, removing the ability to send and receive information with a training server. Furthermore, standard on-board CPUs/GPUs are ill-equipped to handle the training of current state-of-the-art vision models. This need AI models which can be trained on the edge (i.e. close to the data source) has driven interest in more efficient hardware architectures for AI. While more efficient digital architectures optimized for ML are a major industry focus, more exotic approaches beyond the realm of digital computing have also been proposed.

1.2.2 Neuromorphic Acceleration

One computing paradigm that has gained traction in the age of deep learning, *neuromorphic computing*, aims to engineer chips inspired by how computation is done in the human brain. In contrast to traditional digital computing, many neuromorphic architectures additionally leverage analog computing to more closely mimic brain functions, with the idea that hardware more closely aligned with human brains will be faster and more energy efficient for ML than current approaches. Proposals for neuromorphic architectures include optical [93], electronic [118] or even mechanical [55] systems for neural network computation. Most work in this area has targeted building chips for inference only, given the challenge of implementing backpropagation through unconventional hardware. One backpropagation-less approach to machine learning which has gained traction for the potential of hardware acceleration is *reservoir computing* (RC). Chapter 2 of this thesis will further discuss adapting reservoir computing for computer vision.

1.3 Learning Algorithms for Perception

For robotics and autonomous driving applications, a variety of machine learning tasks are required, such as image classification, object detection, semantic segmentation, pose estimation, scene reconstruction, etc. This thesis will focus primarily on two of these: image classification and object detection. *Image classification* is the task of prediction a single class label from a given preset which best describes the image as a whole. *Object detection* is the task of localizing and classifying all objects of interest within a scene. 2D object detection describes objects with four degrees of freedom (**x**, **y**, **height**, **width**), whereas 3D object detection does so with seven degrees of freedom (**x**, **y**, **z**, **height**, **width**, **length**, **heading**).

Within the past decade, advancements in deep learning have propelled neural networks to become the dominant approach for most computer vision problems. For vision-based tasks, convolutional neural network (CNN) [29] and transformer [109] architectures have become the standard backbone for most use-cases. Task-specific variations of these architectures are common for different tasks, e.g. Faster R-CNN for object detection [89] and Mask R-CNN for image segmentation [28]. Supervised learning, in which large amounts of curated labeled data are used to train the model, is the most straightforward approach to training these vision models.

1.4 Data for Perception

Huge advancements in machine learning over the past decade are owed not only to advances in algorithms and compute power, but also to increasingly large troves of training data. Some concrete examples include Stable Diffusion, a cutting-edge text-to-image generative model trained on approximately 5 billion captioned images [90], and Segment Anything, a foundation model capable of zero-shot image segmentation trained on 11 million images and over a billion segmentation masks [42]. A summary of the training data used for these foundation models is included in Fig. 1.3. Supervised learning demands that training data have data labels associated with each data sample. For training vision models capable of semantic understanding, web crawling yields billions of images readily captioned by humans, providing an excellent source of semantic supervision.

Collecting data for 3D perceptions tasks constitutes a significantly bigger challenge than doing so for models trained only for semantic understanding or 2D geometric recognition. Visual data on the web by and large does not contain detailed spatial and geometric descriptions, which are key to training AI models capable of 3D perception. Directly collecting the required data on one's own is not a straightforward solution. While autonomous driving companies with large fleets can record many hours of driving scenes, the process of manually labeling the collected data with human labor is both very expensive and time consuming. In contrast to labeling 2D images with object box labels, 3D labeling requires parsing point clouds to properly localize objects in 3D space, a more niche skill set that cannot be easily crowd-sourced.

Several approaches exist to try and mitigate the data issue. Transfer learning tries

	Large Language Model	Text-to-image Generation	Image Segmentation
Model	GPT−3	Stable Diffusion	Segment Anything
Training Data	>400 billion words	5 billion images/captions	11 million images, >1 billion masks

Fig. 1.3: Comparison of training data used for training different foundation models.

tries to adapt models trained on a different domain for a new task (e.g. leveraging image foundation models for point cloud perception). Self-supervised learning uses purely unlabeled data during training to teach a model to learn good representations through forcing the model to recognize intrinsic patterns in the data. Self-supervised learning is particularly useful in training language models, as text data can easily be used to provide its own supervisory signal. Weakly-supervised learning tries to train models using some low-quality, noisy form of annotation. Semi-supervised learning tries to train a model using a small corpus of labeled data, combined with a large reservoir of unlabeled data. A popular approach to semi-supervised learning is *pseudolabeling*, in which a model is trained in a supervised manner on the labeled data and thereafter used to generate predictions on the unlabeled data to be used in downstream training. This approach will be explored in more detail in chapters 4 and 5 of this thesis.

1.5 Outline of the Dissertation

This dissertation summarizes our efforts towards addressing all of the aforementioned problems of AV perception. Chapter 2 introduces a novel form of hardware acceleration, which we dub a hybrid form of reservoir computing, capable of accelerating the training of neural network models in an edge environment. Chapter 3 addresses the problem of sensor fusion, and proposes a novel approach for LiDAR-camera fusion which is both accurate and low-latency. Chapters 4 and 5 explore the problem of label-efficient 3D object detection, and propose both a theoretically-grounded approach to improved pseudolabeling, and an approach leveraging the trajectory prediction portion of the autonomy stack to improve perception. Lastly, chapter 6 provides a summary and future outlooks.

Chapter 2

Reservoir Computing for Accelerated Computer Vision

In this chapter we will address the problem of efficient hardware architectures to enable on-board training of computer vision models. Specifically, we introduce a novel form of reservoir computing that is adapted for image classification tasks and is able to achieve comparable performance to that of CNNs with greatly reduced training time. This chapter is based on research previously published in [33] and [34].

2.1 Reservoir Computing Introduction

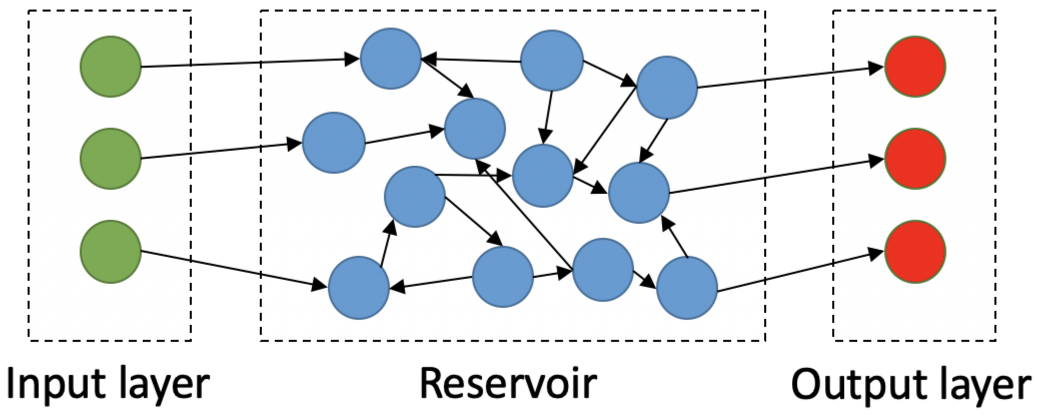


Fig. 2.1: Diagram of prototypical RC scheme. Input data is encoded via the input layer, before the reservoir performs a transformation into a high dimensional space. Lastly, the output layer converts the reservoir readout to the target sequence.

Reservoir computing is a framework for machine learning drawing inspiration from traditional Recurrent Neural Networks (RNN), a variant of artificial neural networks which include recurrent connections between nodes in the hidden layers [37, 71, 36]. The prototypical RC network is composed of three components: an input layer, a "reservoir", and an output layer. The input layer linearly transforms the input signal with a set of fixed random weights to be fed into the reservoir. The key part of RC, the reservoir, is comprised of a series of nonlinear nodes or neurons with random connections in between such that recurrent loops exist within the reservoir (hence mimicking the internal dynamics of an RNN). The last layer of the network, the output layer, is a single linear layer which transforms the reservoir output into the target sequence. Fig 2.1 includes a diagram of this generalized structure of RC. The weights associated with the input layer and reservoir are randomly initialized and remain fixed during training; only the output layer weights are trained using a standard linear regression. By eliminating the need to train the hidden layers of the neural network via backpropagation, reservoir computing vastly reduces the amount of computation required for training while also guaranteeing convergence.

Based on the massive reduction in number of trainable parameters, it's not immediately obvious why RC should be able to perform competitively with standard neural networks. The key to success is in the choice of a reservoir: a successful reservoir should transform the input data into a high-dimensional feature space. As a nonlinear transformation of sample data from a low dimensional space into a high dimensional space increases the likelihood of linear separability, a successful reservoir improves the classification power of the trained linear output layer. An example of this effect is shown in Fig. 2.2. Furthermore, the dynamics of the reservoir should be reproducible, i.e. for a given input signal, the reservoir should respond in the same way every time.

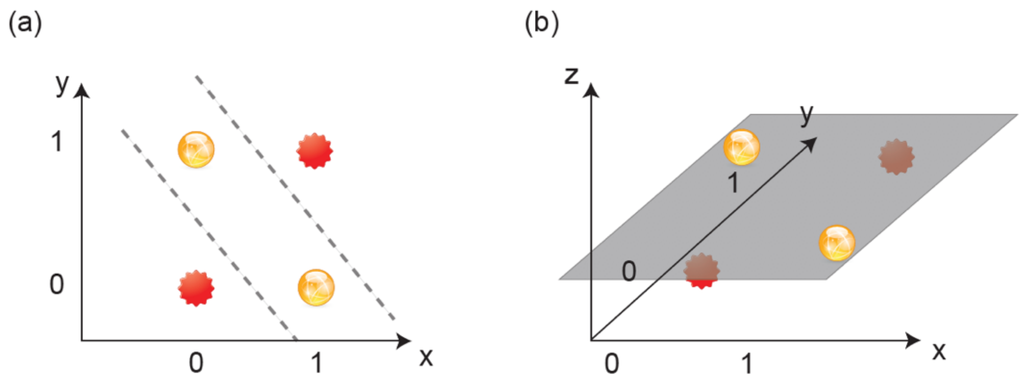


Fig. 2.2: Illustration of linear separability arising from a nonlinear transformation to a higher dimension. **(a)** Two classes lying in a 2D space which are not linearly separable. **(b)** After a nonlinear transformation into 3D space, the two classes are now separable with a single linear hyperplane. Adapted from [4].

The few constraints required of a serviceable reservoir allow for a variety of methods of implementation, including both software simulations and physical manifestations. Successful reservoir computers have been built with methods as exotic as cell cultures [25], water tanks [72], and octopus arms [75]. More practical approaches commonly leverage some form of analog electronics due to the maturity of CMOS manufacturing processes. Nonetheless, physical implementations of reservoirs are bogged down by the requirement that each node in the reservoir requires a physical nonlinear element. While implementation is feasible for reservoirs on the order of hundreds of nodes, scaling to the number of neurons in modern machine learning applications (on the order of millions) becomes prohibitive due to size and energy consumption constraints. To address this problem, an alternative formulation known as *delay-based* RC was proposed

2.2 Delay-based RC

Delay-based RC is an alterntive to classical RC that has gained significant traction recently [4]. As opposed to a spatial reservoir, delay-based RC instead uses a fully temporal reservoir, where physical nodes are replaced with so-called *virtual nodes*, created through time division multiplexing. These virtual nodes are processed serially through a single physical node with delayed feedback; more virtual nodes can be stored in the system memory through simply increasing the length of the delay. The key insight of delay-based RC is that time-delay dynamical systems exhibit a high-dimensional state space, meeting the requirement for a suitable reservoir. From an implementation perspective, delay-based RC significantly reduces the number of required components, making it an attractive candidate for hardware acceleration. An outline of delay-based RC is described in the following section.

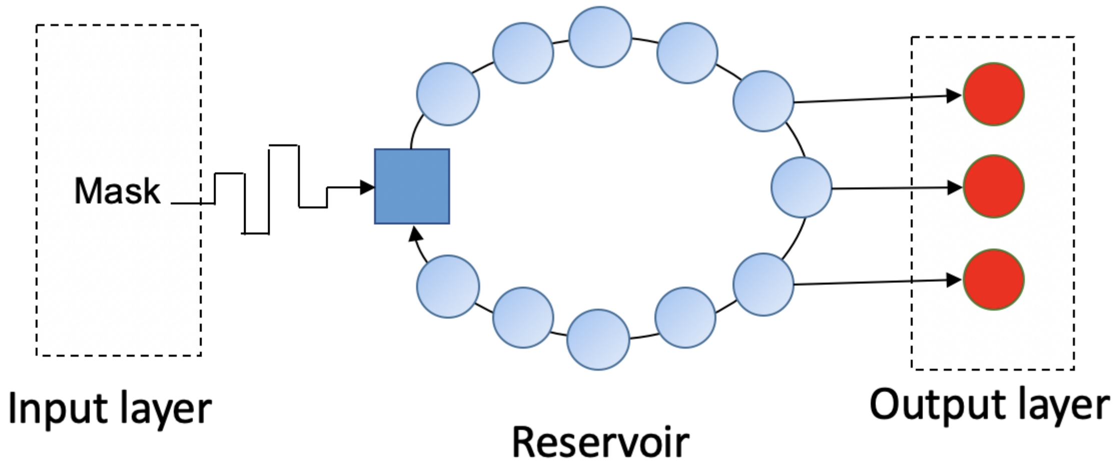


Fig. 2.3: Diagram of time-delay based reservoir computing scheme. Virtual nodes are created through multiplying input data by mask matrix, which are then serially passed through nonlinear node with delayed feedback. Reservoir output is read through sampling in time.

2.2.1 Conceptual Outline

The fundamental change in the delay-based formulation is a replacement of physical nodes in a spatial reservoir with virtual nodes, which exist as the system response at a given time. Given a data sample with some number of feature channels for input to the reservoir, we first serialize the data into a stream $I(t)$ using a sample-and-hold operation. The input layer of the RC model takes the data stream $I(t)$ and converts it into a discrete set of virtual nodes $u_{in}(t)$ through a process called *masking*. Masking is done using a matrix M, known as the mask, with dimension $N \times K$, with N being the number of desired virtual nodes and K being the number of input feature channels. These N virtual nodes are fed into the reservoir, consisting of a nonlinear element with some associated delay line of length τ_D. As τ_D corresponds to the total effective memory of the reservoir, the virtual nodes are fed in sequentially with spacing $\theta = \tau_D/N$, such that all virtual nodes corresponding to a single sample fit into the system memory at once. Fig. 2.4 illustrates in detail the operation of inputting data to the reservoir. Given this setup, the reservoir response, $x(t)$, is described by:

$$T\frac{dx(t)}{dt} + x(t) = \alpha f(x(t-\tau_D) + u_{in}(t) + \beta) \tag{2.1}$$

where f is the nonlinear activation of the physical node, τ_D is the delay time, α is the gain of the entire circulation through the delay line, $u_{in}(t)$ is the masked input data, β is a constant bias, and T is the characteristic time of the system corresponding to the maximal time between which neighboring virtual nodes can interact. The continuous reservoir response is sampled discretely to generate the virtual node values, $X(k)$, where k denotes discrete time. The final output layer is a linear mapping of the virtual node values to the desired predicted labels, $\hat{Y}$:

$$\hat{Y} = W^{opt} X(k) \tag{2.2}$$

where W^{opt} are the trained readout weights. For classification-type tasks, the true class labels Y are encoded through one-hot encoding; i.e. Y is an $N_{samples} \times N_{classes}$ matrix, with a binary encoding used to denote the true class of a given image. Classification of the predicted labels $\hat{Y}$ is done with a winner-take-all approach, in which the class with the highest value is chosen to be the model prediction. Training is done using the ridge regression routine [104], where the equation for W^{opt} is given by:

$$W^{opt} = \underset{W}{argmin} \, ||Y - WX(k)||^2 + \lambda ||W||^2 \tag{2.3}$$

where λ is a regularization constant used to prevent overfitting. This optimization is performed offline, after all of the input data has been circulated through the reservoir.

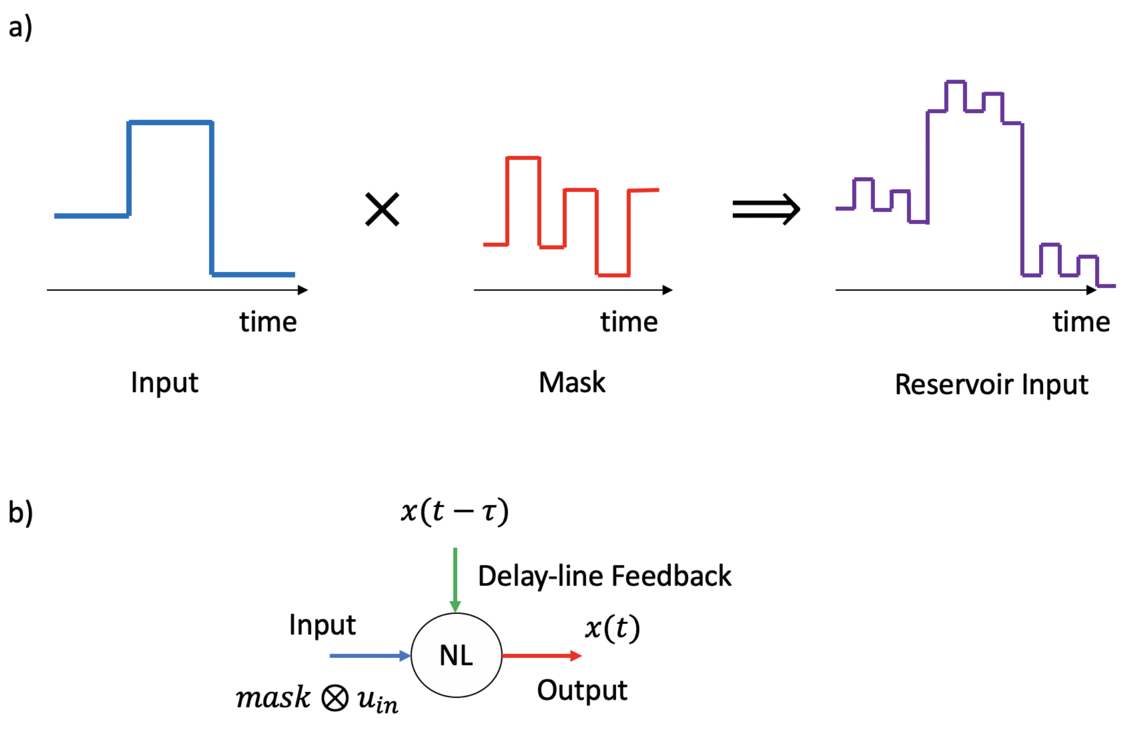


Fig. 2.4: **(a)** Visualization of input masking operation, demonstrating multiplication of input sequence with a random mask. **(b)** Detailed illustration of inputs and outputs to the nonlinear element.

2.2.2 Optoelectronic Hardware Implementation

Optics is an attractive medium for building neuromorphic hardware accelerators due to the potential of fast processing speeds and low power consumption. Reservoirs built with photonic integrated circuits remain relatively small in scale (limited to tens of nodes)

[107, 26, 101], whereas using free space optics, though able to incorporate a larger number of nodes, sacrifices the compactness of an integrated solution [3, 1].

Optoelectronic [78, 50, 49] implementations of delay-based RC have emerged as a happy medium, achieving strong performance on several benchmark tasks, such as time series prediction and voice recognition. A well-known implementation follows the optoelectronic oscillator (OEO) model [50, 126]. In the OEO model, light from a laser is modulated by an electro-optic modulator before passing through an optical fiber line. After the time delay, the optical signal is converted to the electrical domain via photodiode, amplified, and then fed into the drive port of the modulator. Fig. 2.5 contains a visualization of the optoelectronic RC architecture.

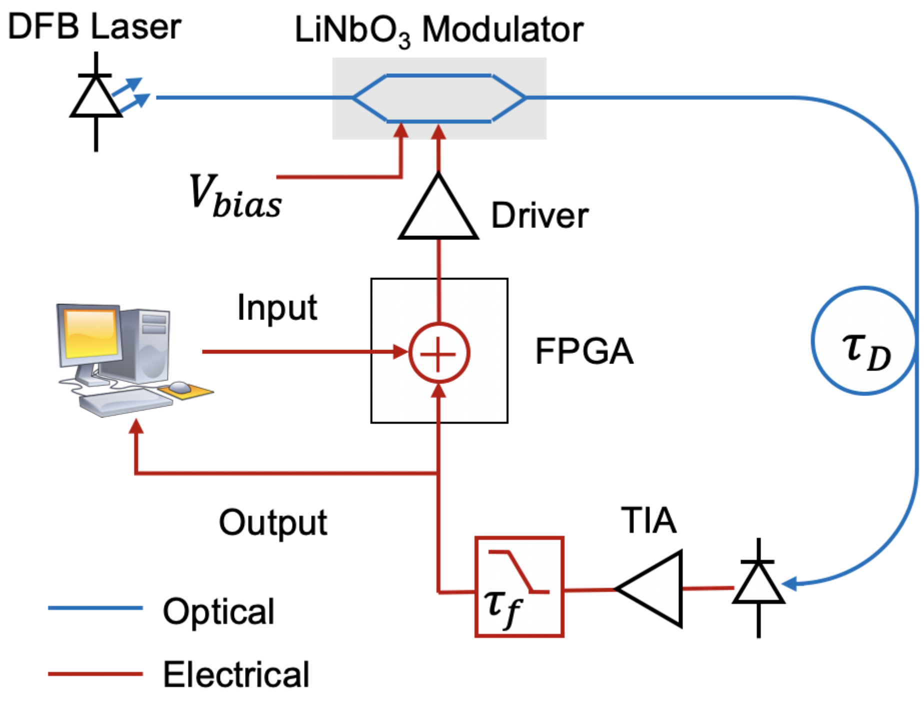


Fig. 2.5: Diagram of time-delay based reservoir computing scheme. Virtual nodes are created through multiplying input data by mask matrix, which are then serially passed through nonlinear node with delayed feedback. Reservoir output is read through sampling in time.

2.2.3 Limitations

Although delay-based RC is attractive from a hardware acceleration perspective, the relative structural simplicity arising from the lack of full spatial coupling (i.e. lack of connectivity between all nodes in the reservoir) makes scaling delay-based RC to tackle more complex tasks challenging. Adapting reservoir computing for processing images, a necessary perquisite for deployment in AV perception, is a challenging task.

Image recognition, a task central to computer vision and well-representative of tasks required in AV perception, has remained relatively under-explored in the RC literature because of these inherent limitations. With recent advancements in deep learning, classical image processing techniques using hand-crafted features have ceded ground to convolutional neural networks (CNNs), which have been able to achieve state-of-the-art performance on most tasks [44, 53]. CNNs are a class of neural networks which convolve input images with trained filters to extract useful features for classification, allowing the neural network to learn on its own the most useful features, rather than relying on more traditional feature engineering.

In contrast to CNNs which exploit the structure of images with local receptive fields in each layer, reservoirs contain largely random connections between interior nodes, limiting the ability of the model to capture spatial dependencies within images. Fig. 2.6 compares the more structured convolutional feature extraction to that of a reservoir. Delay-based RC somewhat limits the structural randomness of the reservoir interior by restricting interactions to between nodes close together in time, however still lacks the feature extraction capabilities of a CNN. The next section of this chapter describes our proposed scheme for adapting delay-based RC to be capable of handling image recognition tasks.

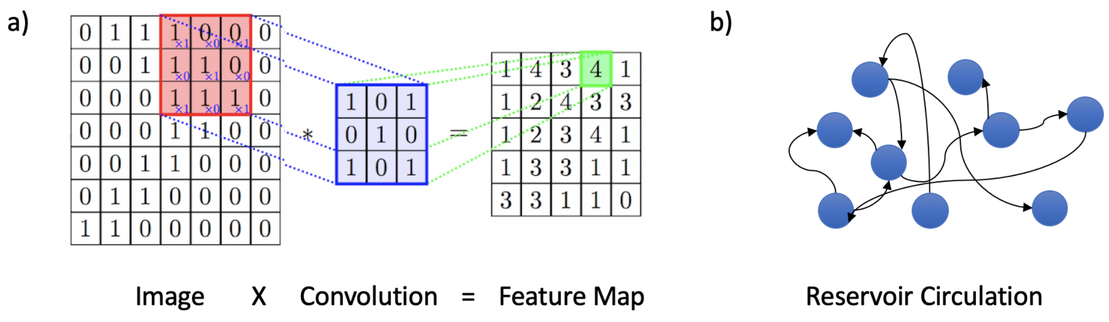


Fig. 2.6: **(a)** Feature extraction using convolutional layer, illustrating local receptive field. **(b)** Random connections inside reservoir, poor contrast to CNN for processing images.

2.3 Hybrid Convolutional RC

In this section, we introduce our adaption of delay-based optoelectronic RC for image classification, which we dub hybrid convolutional RC.

2.3.1 Overview

The key insight of our proposed hybrid RC scheme is to try and capture the power of convolutional feature extraction and integrate it into the RC pipeline. In this vein, we have introduced a new form of RC which combines standard delay-based RC with preprocessing images through convolutional layers. After passing through the series of convolutional and pooling layers, the output feature maps are flattened row-wise into a single vector, which is then multiplied by the random mask matrix to generate the input nodes. Thus, each input node becomes a random linear combination of the output convolutional features, the exact nature of which depends on the mask matrix. Normally, the weights within convolutional layers are trained through backpropagation to allow the CNN to learn the most optimal features for classification. We consider two models for our hybrid scheme: one which maintains the training of convolutional layers through backpropagation, and another in which we instead use convolutional layers which are randomly initialized, but then left untrained. The latter method, instead of extracting learned features, generates randomized feature maps to preserve the fast training speed of RC. The number of feature maps generated is controlled through the final convolutional layer's width. After passing through these layers, all of the feature maps are flattened into a single feature vector and multiplied with a mask matrix before being processed through the reservoir computer itself. Fig. 2.7 compares our proposed hybrid RC approach with CNNs and the standard delay-line RC approach. The following sections will discuss in further detail the implementation of our proposed system.

2.3.2 Convolutional Preprocessing and Masking

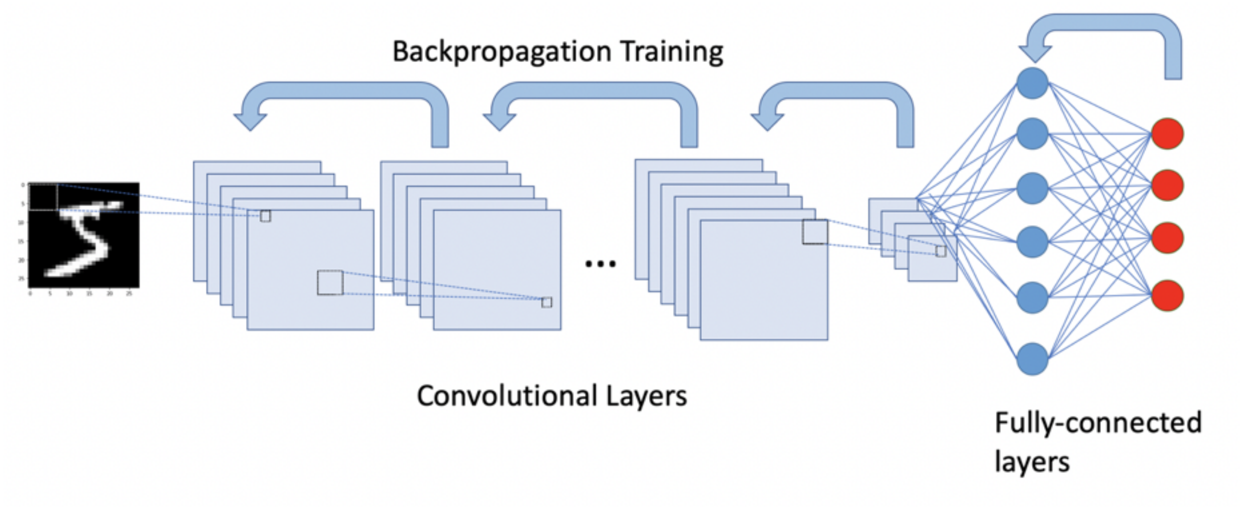
The first step of our proposed hybrid RC scheme is preprocessing through a set of convolutional layers. The convolution of an input image X with a set of filter weights w is represented with the following equation:

$$Z_{u,v,n} = \sum_i \sum_j \sum_c w_{i,j,c,n} * X_{u-i,v-j,c} \tag{2.4}$$

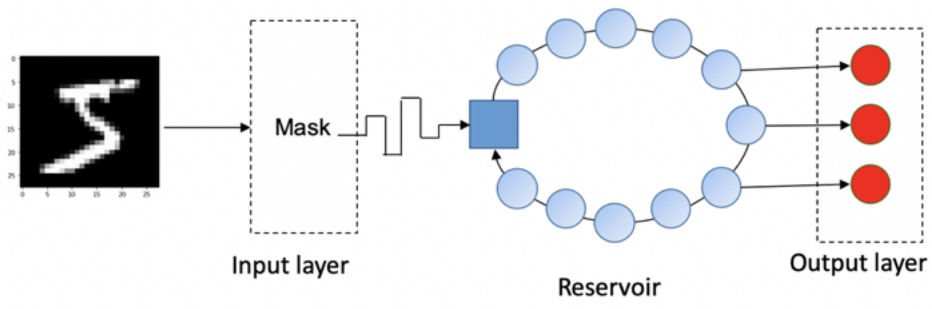
where c corresponds to the number of input feature channels, and n corresponds to the number of output feature channels. After the forward pass through the convolutional layers, the encoded image is flattened into a single feature vector, first along the channel dimension, then along the two spatial dimensions. Next, the feature map outputs are transformed into the input virtual nodes through the masking process. We use a two-dimensional mask matrix with size $N_{virtual\ nodes} \times N_{feature\ map\ dimensionality}$, which is matrix-multiplied with the flattened feature vector to generate the virtual nodes. Through adjusting the number of rows of the mask matrix, we can adjust the number of virtual nodes to be generated per-image. The mask matrix itself is a sparse random matrix, with the sparsity of the random matrix treated as a hyperparameter. Thus, each input virtual node becomes some random linear combination of pixel-level features.

2.3.3 Hardware Implementation

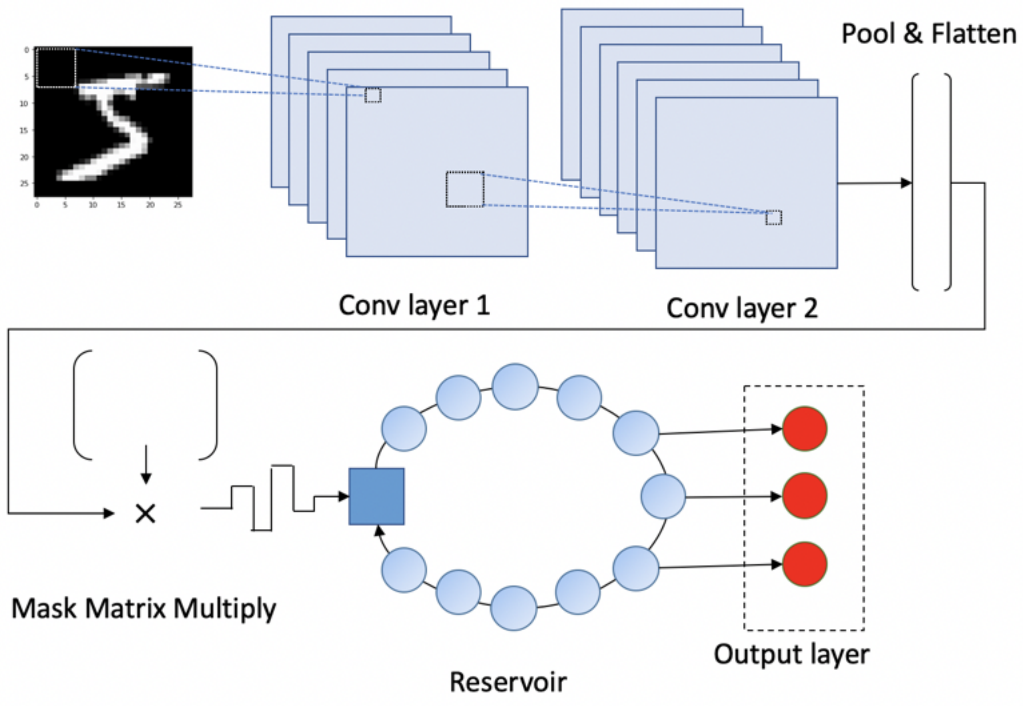
To implement our proposed hybrid RC scheme experimentally, we adapt the OEO-based reservoir computer with FPGA control, and use a standard laptop computer for both data preprocessing and postprocessing. First, images are encoded with the convolutional layers and then undergo the masking process digitally on the laptop. Next, the reservoir input is transferred via ethernet connection to an FPGA with on-board ARM processor directly interfacing with the analog system. The input data is converted to an analog signal via an DAC and circulated through the reservoir, after which the reservoir response is measured and converted back to a digital signal via an ADC. The output data is then transferred via ethernet back to the laptop computer, where the final step of linear regression is performed. An overview of this experimental procedure is shown in Fig. 2.8. Each individual aspect of the hardware implementation is described below.



(a)



(b)



(c)

Fig. 2.7: (a) Prototypical CNN architecture with illustration of backpropagation training. **(b)** Standard delay-based RC architecture. **(c)** Architecture of our proposed hybrid RC scheme, illustrating preprocessing through untrained convolutional layers followed by delay-based RC.

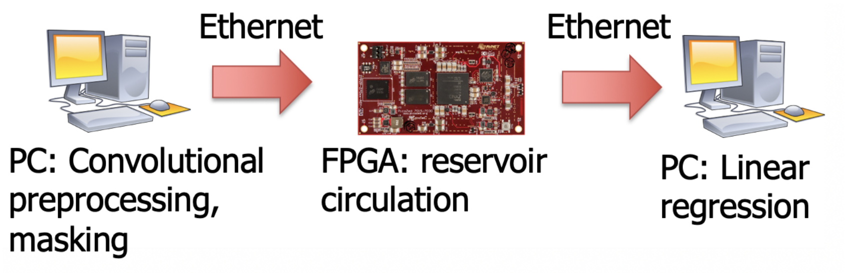

Ethernet
Ethernet
PC: Convolutional preprocessing, masking
FPGA: reservoir circulation
PC: Linear regression

Fig. 2.8: Illustration of our RC experimental procedure. Experimental data is preprocessed digitally on a PC before transfer via ethernet to an FPGA for reservoir circulation. The reservoir readout is transfered back to the PC via ethernet for the final training step.

2.3.3.1 Computer for Digital Processing

The first two steps in our hybrid RC scheme, the forward pass through convolutional layers and the input masking operation, are performed digitally. Although our FPGA includes an on-board ARM processor capable of performing these operations, for the sake of simplicity we perform these off-board on a laptop computer, although we submit that a more compact implementation can make use of the on-chip CPU for all digital tasks. We implement our preprocessing scheme in Python using the Keras and numpy libraries. The computer used is a MacBook Pro 2.6 GHz 6-Core Intel Core i7.

2.3.3.2 Optoelectronic RC Implementation

We base the delay-line reservoir computer portion of our scheme on the aforementioned OEO implementation shown in Fig. 2.5. The experimental set-up consists of a 1550 nm distributed feedback laser (Gooch & Housego DS-7009) modulated by a 40 Gb/s $LiNbO_3$ Mach-Zehnder modulator (Fujitsu FTM7937EZ). The modulated signal passes through an optical delay line before being converted to the electrical domain with a high-speed photodiode (MACOM P-18A). After passing through a low-noise transimpedance amplifier and low-pass filter, the analog signal is sampled by an analog-to-digital converter (ADC) with 12 bit resolution attached to a field programmable gate array (FPGA) mezzanine card (Zipcores FMC-DSP Rev. B). The low-pass filter here is used to set the effective memory of the delay system, i.e. how far apart in time virtual nodes can still interact with each other. The reservoir response is added digitally to the input signal within the FPGA's programmable logic, before being output through a digital-to-analog converter (DAC) used to drive the modulator input. Fig. 2.9 contains a photograph of our experimental testbed.

The differential equation describing the reservoir response $x(t)$ of this system (a modified version of Eq. 2.1) is:

$$\tau_f \frac{dx(t)}{dt} + x(t) = G_{TIA} sin^2(G_{DRV} x(t - \tau_D) + G_{in} u_{in}(t) + V_{bias}) \tag{2.5}$$

where τ_f is the time constant of the low-pass filter, G_{TIA} is the gain from the transimpedance amplifier, G_{DRV} is the gain from the driver amplifier, G_{in} is the digital amplification of the input signal, and V_{bias} is a DC bias voltage applied to the Mach-Zehnder modulator. We note that the sin^2 nonlinearity comes from the response function of the Mach-Zehnder interferometer. Given the sin^2 nonlinearity, applying a V_{bias} allows us to control the relative strength of the nonlinear response. V_{bias} close to V_π (i.e. voltage corresponding to a π radian shift) puts the modulator in the strongly nonlinear portion of the sine function, whereas a V_{bias} close to $V_\pi/2$ puts the modulator in the approximately linear regime of the sine function.

The bandwidth of this system is limited by the ADC/DAC, which have a 125 MSa/s sampling rate. For the experiments desrcibed in this thesis, we use a virtual node spacing is 80 ns, allowing for a total information processing speed of 12.5 MHz. We base this choice off of limitations of the digital logic needed for transferring data between the chip and PC.

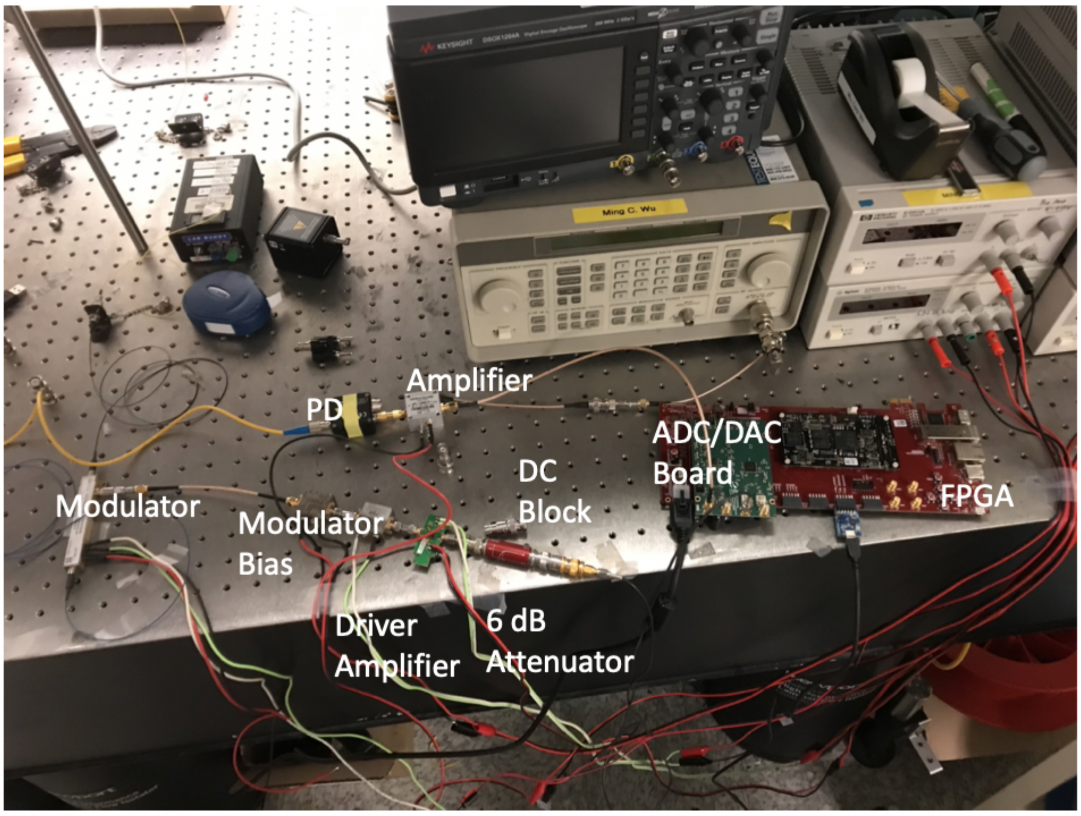


Fig. 2.9: Labeled photo of reservoir computing experimental setup.

2.3.3.3 FPGA Design

To handle data I/O between the analog reservoir and digitally pre/postprocessed data, we use a PicoZed board with a Zynq 7000 system-on-chip (SoC). Fig. 2.10 contains a high-level block diagram of the Zynq 7000. The system-on-chip can be broadly divided into two components. The first of these is the processing system (PS), which contains dual ARM CPU cores and access to large blocks of DRAM memory. The second component, the programmable logic (PL), contains an array of programmable latches that can be used to create custom logic functions. To move data between the PS and PL, the industry-standard AXI protocol is used. Both of these components are used in our hardware design for control of the reservoir computer.

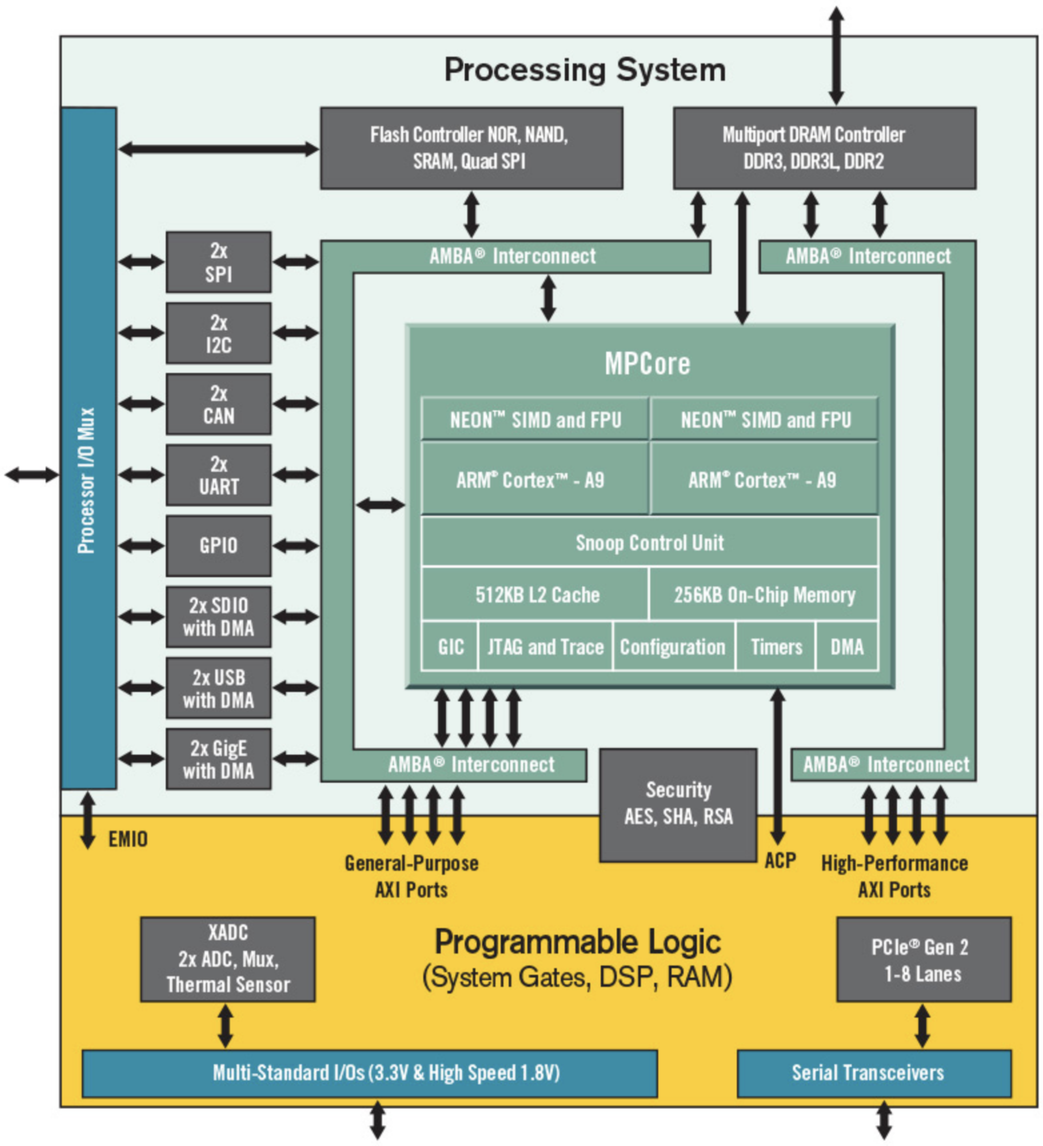


Fig. 2.10: Block diagram of the AMD Zynq 7000 SoC.

The key functionality of the PL in our system is to interface with the ADC/DAC module on the attached mezzanine card. This can be broken down into several sub-processes: receiving input data from the PS, reading the reservoir response through the ADC, performing feedback addition between the reservoir response and input signal (and optionally scaling the signal), outputting the combined signal through the DAC to drive the modulator, and sending the sampled virtual node response back to the PS. Fig. 2.11 contains a high-level illustration of the tasks requiring custom programmable logic.

To program the PL, we use Xilinx's Vitis and Vivado software kits. Within these, Xilinx provides an IP block integrator, allowing us to leverage pre-built components for transferring data between the PS and PL. Thus, to integrate our custom functionality into

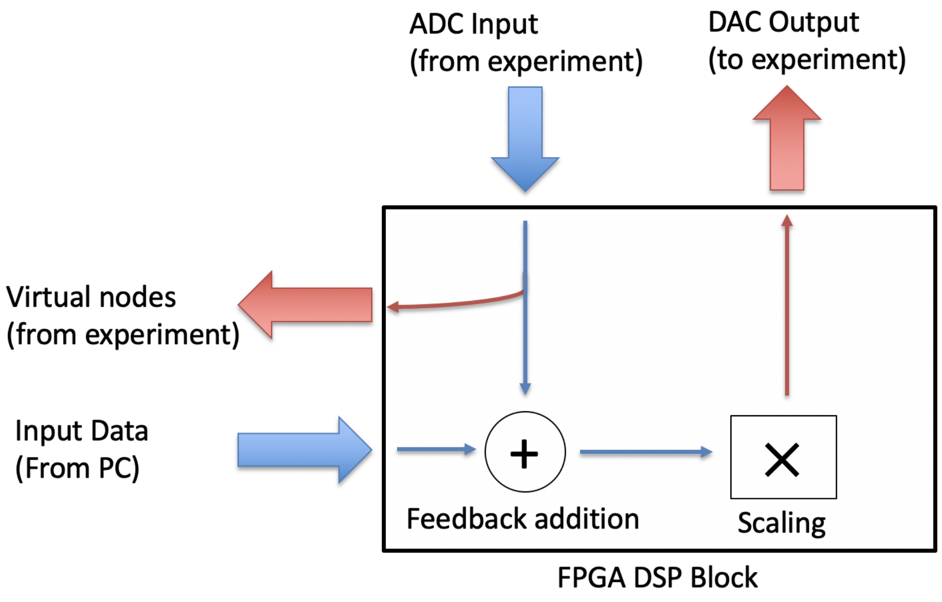


Fig. 2.11: Illustration of tasks performed by custom FPGA DSP block used for experiment control.

the high-level block design, we create a custom DSP block written in Verilog containing the core functionalities described above and then compile it into a module compatible with the block-level IP integrator. To physically move data on-board, we use an AXI DMA (direct memory access) block, a data moving engine capable of moving data between the PS and PL. In addition, we connect FIFO blocks on either end of our custom DSP block to queue departing and arriving data. Finally, all of the described components are connected via the AXI interface, enabling data movement between all parts of the SoC. An example of the Xilinx interface with connected block components is shown in Fig. 2.12.

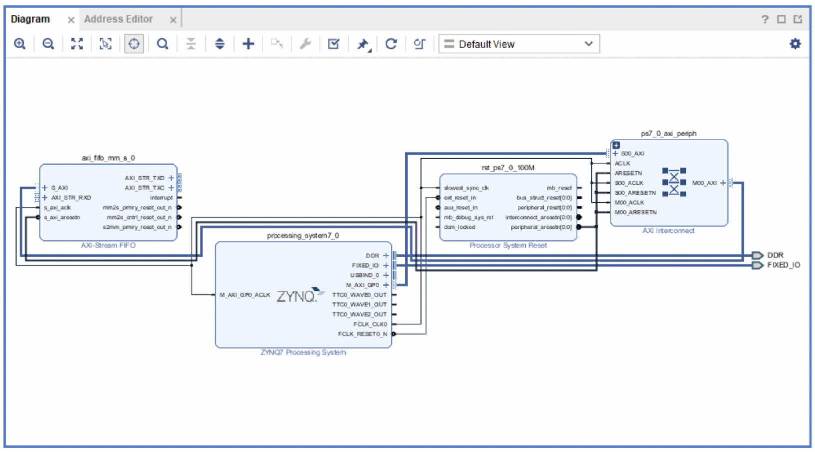


Fig. 2.12: Screenshot of Xilinx IP Integrator interface with AXI DMA and FIFO blocks.

The critical function of the on-board PS is sending and receiving data between the board and the PC. To enable experiments at high speed, we use the Ethernet protocol for transferring data between the PC and board at speeds of up to 1 Gbps. We program the PS using the lwip (Lightweight IP) library, which gives networking functionality for bare metal programs (we program the PS without an OS for simplicity sake). Ethernet packets arrive from the PC in groups of 1446 bytes (largest size for standard Ethernet packet). After arrival, the input data is parsed and stored in the DRAM memory of the PS. The DMA reads data from the memory in blocks of 4 MB to transfer to the PL for reservoir circulation. Although the DMA can move data in larger size blocks, we find empirically that 4 MB blocks offer the optimal processing speeds due to memory overhead associated with larger packets. After the reservoir response is sampled, the DMA moves a corresponding 4 MB packet back to the PS, where it is once again placed into DRAM memory. The PS then reads these output virtual nodes from memory, groups them into Ethernet packets, and sends them out from the board back to the PC. Repeating this workflow for groups of 4 MB of data at a time, we can scale our data transfer operation to process an arbitrarily large number of input virtual nodes. Although speeds of up to 1 Gbps are theoretically possible for Ethernet protocol, we observe speeds closer to 250 Mbps, as the use of jumbo Ethernet packets (>1446 bytes) is required to fully saturate the link. As jumbo packets are not compatible with our PicoZed model board, we operate the Ethernet connection below its limit. On the PC side, we use Socket, a Python API capable of sending and receiving data via Ethernet connection. The overall design of both the PL and PS components of the SoC are detailed in Fig. 2.13.

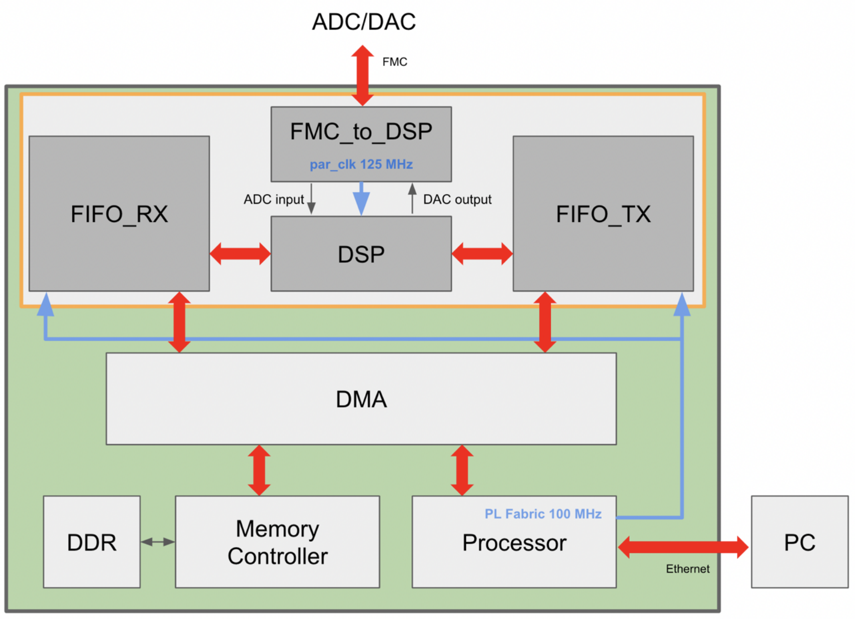


Fig. 2.13: Diagram detailing the overall FPGA configuration used in our experiments.

2.3.4 Simulation Software

In addition to our hardware testbed, we also develop simulation software to model our hybrid RC scheme digitally. Our simulation model uses Euler's method to numerically solve the differential equation in Eqn. 2.5. In addition to the physical constants contained within the equation for the system dynamics, we also simulate the 12 bit resolution arising from the discretization performed by the ADC/DAC. To optimize all of the experiment hyperparameters, we perform a standard grid search to find a good operating point.

2.4 Experimental Results

To experimentally demonstrate our proposed RC scheme, we use the well-known MNIST handwritten digit classification task [52]. The MNIST dataset consists of 70000 (60000 training, 10000 test) grayscale 28×28 images of handwritten digits 0-9; the goal of the task is to correctly identify the written number. We consider two benchmarks for this task: first the LeNet-5 CNN, a well-known CNN architecture adapted for this problem, which achieves a 0.95% test error [53]. Modern neural networks with even deeper architectures have further improved accuracy on MNIST to a few fractions of a percent [18], however these exceed the training capabilities of a standard CPU and so are less useful to us as benchmarks. Additionally, we also consider an Extreme Learning Machine (ELM), a variant of neural network containing layers of untrained neurons, as a benchmark [32]. We use an ELM containing two convolutional layers and one fully-connected layer with 15000 neurons, all of which are left untrained, to serve as a memoryless corollary to our hybrid RC scheme. This benchmark achieves a 1.4% test error on the MNIST task.

2.4.1 Parameter Optimization

Our model has several adjustable parameters that need to be optimized, both physical and digital. We explore these primarily through experiments, running scaled-down experiments of 1000 virtual nodes per image to measure performance.

2.4.1.1 Convolutional Preprocessing

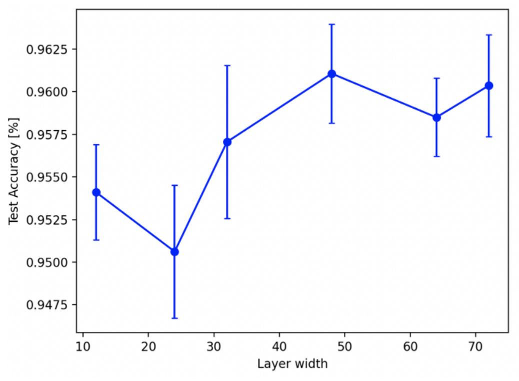
Within the convolutional layers used to generate input feature maps, there are several parameters we can adjust: the number of layers, the width of the layers, convolutional kernel size, and the inclusion of pooling layers. Unlike in trained CNN models, performance improvements from adding more untrained convolutional layers level off relatively quickly (see Tab. 2.1); for this task, we use only two layers. The width of the layers, representing the number of convolutional kernels included within the layer, similarly has a small effect on performance once a small threshold has been passed. Fig. 2.14 shows the results of our experiments varying the convolutional layer(s) width. In our experiments, we use layers of width 32 and 64, respectively. We use a 3×3 convolutional kernel within each of these layers, consistent with many modern CNN models [97]. Additionally, we include a third 2×2 max-pooling layer as part of this preprocessing scheme, as we found it to significantly increase test accuracy while further reducing dimensions and processing time, as shown in Tab. 2.2. Processing images through these three layers produces a flattened vector of dimension 9216 x 1.

2.4.1.2 Random Matrix Masking

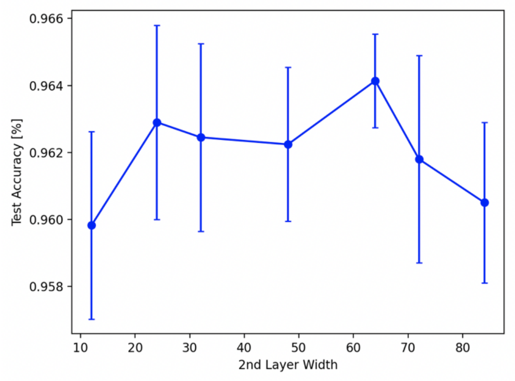
The random mask matrix, used to generate the input nodes to the reservoir computer from the preprocessed feature maps, has several adjustable parameters associated with it. The first of these, the matrix density, denotes the number of non-zero elements in the matrix,

Number of Convolutional Layers	Classification Accuracy
0	88.039±0.75%
1	96.107±0.28%
2	96.41±0.14%
3	96.215±0.36%

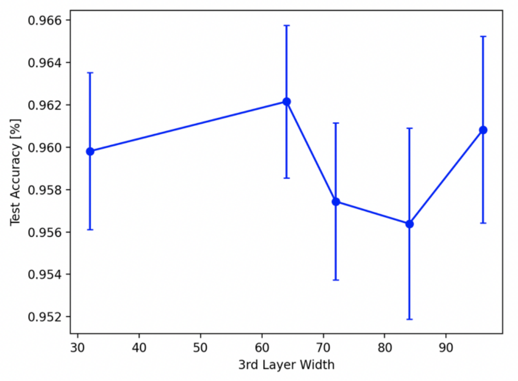
Table 2.1: MNIST classification accuracy varying number of layers in convolutional preprocessing stage.



(a)



(b)



(c)

Fig. 2.14: Accuracy on MNIST test set for varying the convolutional layer width of the (a) first (b) second and (c) third layer. Overall, no significant trend is observed.

Number of Convolutional Layers	Classification Accuracy
w/ pooling	96.41±0.14%
w/o pooling	95.178±0.5%

Table 2.2: MNIST classification accuracy with and without max pooling layer after convolutional preprocessing.

representing the fraction of matrix elements with non-zero values. For this problem, a low matrix density of 0.01 is found to achieve optimal experimental performance (see Fig. 2.15). The other parameter associated with the mask matrix is the random distribution from which the non-zero elements are drawn from. A summary of experiments using differing random distributions is shown in Tab. 2.3; we found the optimal to be a uniform random distribution with range $(-1, 1)$. After the matrix is initialized, we perform a row-wise normalization to limit the range of generated input nodes.

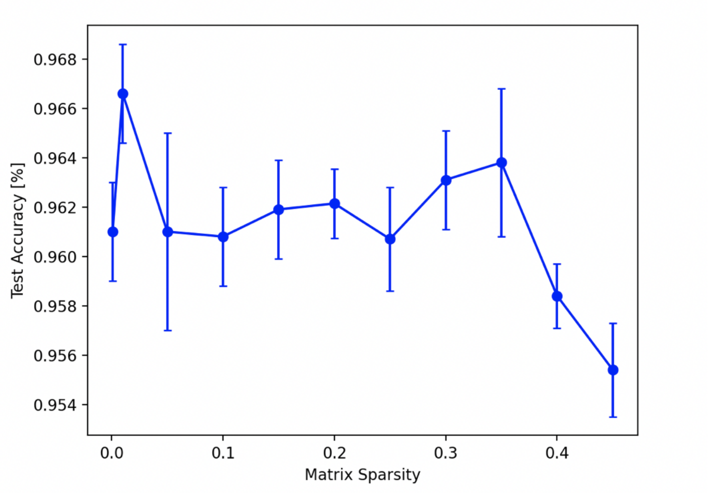


Fig. 2.15: Accuracy on MNIST test set with varying sparsity of mask matrix.

Random Distribution	Classification Accuracy
Uniform [-1,1]	96.66±0.24%
Binary (-1,1)	95.96±0.29%
Normal σ=2	96.41±0.14%
Normal σ=1	96.37±0.27%
Normal σ=0.5	96.03±0.25%

Table 2.3: MNIST classification accuracy for differing random distributions used in mask matrix.

2.4.1.3 Experimental Parameters

In addition to the parameters associated with preprocessing, there are also several experimental parameters to be optimized in our optoelectronic experimental testbed. First, we consider the delay-line length and the low pass filter constant. Although traditionally the delay length is chosen such that all of the virtual nodes for a single data sample can fit entirely in memory, we note that it's possible to store virtual nodes for a single sample across multiple delay lengths by simply allowing the reservoir to continue circulating. We pursue this approach for the practical reason of requiring a more manageable length of optical fiber. Because the MNIST task does not inherently require memory, we found that experimental performance has a fairly weak correlation with the delay-line length, with test accuracy obtained using a 100 meter and 2 meter delay being equivalent, as long as laser power is increased to compensate for loss in the longer delay line. For convenience, we use the short 2 meter delay in the rest of our experiments. To validate the idea of using an RC (i.e. memory-dependent) approach for a static image recognition task, we also tried eliminating the system memory by simply removing the feedback addition. In this case, we found a sizable decrease in test accuracy of 0.7% from experiments in which memory is included. Thus, while the inclusion of a memory mechanism adds a small boost to performance from the introduction of a further (random) degree of freedom, storing a larger number of virtual nodes in a long delay-line has a minimal effect. This result is as we expect, since individual virtual nodes have little spacial relation between each other due to the random masking process, meaning a large memory bank does not offer a more meaningful global view of the image. While the delay length controls the total memory storage available, the low pass filter constant determines the temporal distance between which virtual nodes have connections. We use a low pass filter with a time constant approximately equal to one times the virtual node spacing; longer time constants result in degrading performance as a result of too much averaging (see Tab. 2.4) [4].

Low Pass Filter Cutoff Frequency	Classification Accuracy
32 MHz	$96.09 \pm 0.40\%$
100 MHz	$95.35 \pm 0.18\%$

Table 2.4: MNIST classification accuracy for differing low pass filter cutoff frequencies.

Also within our experiment are several parameters used to control the operating point of the dynamical system itself (i.e. whether the system is in an oscillatory regime or not). In the OEO model, these are typically captured by the single-loop gain, i.e. the small-signal gain of a single circulation through the entire system, and the modulator bias point [126]. The former can be most directly controlled through the power of the laser source, whereas the latter is simply the DC bias voltage of the Mach-Zehnder Modulator, in terms of V_π, with $1V_\pi$ corresponding to the minimum transmission of the modulator. We performed an experimental parameter sweep of these two quantities, illustrated in Fig. 2.16. We find that test accuracy is generally stable, except in the quadrant corresponding to high laser power and low modulator bias. This part of the parameter space is congruous with the OEO's oscillation regime, illustrating the deleterious effect of the induced oscillation on the system's performance. Thus, for this task, we can choose the system's operating point fairly liberally while maintaining fairly consistent performance. For the experiments in this paper, we used a modulator bias of $1V_\pi$ and a laser power of 3.5 mW, while driving the modulator at 0.4V.

2.4.2 Untrained Hybrid RC

First, we discuss experimental results using untrained convolutional layers for our hybrid preprocessing scheme. The main appeal of this approach is the minimization of processing time; by using completely untrained convolutional layers, the preprocessing stage is reduced to the time required to randomly initialize the weights within the layers, followed by performing a single forward pass. In this scheme, we randomly initialize the weights using TensorFlow's `glorot_uniform` initializer, which draws weights from a random uniform distribution whose interval is inversely proportional to the sum of the input and output dimensions of the layer. A visualization of some examples of the randomized features produced with this scheme is shown in Fig. 2.17. We performed simulations and experiments

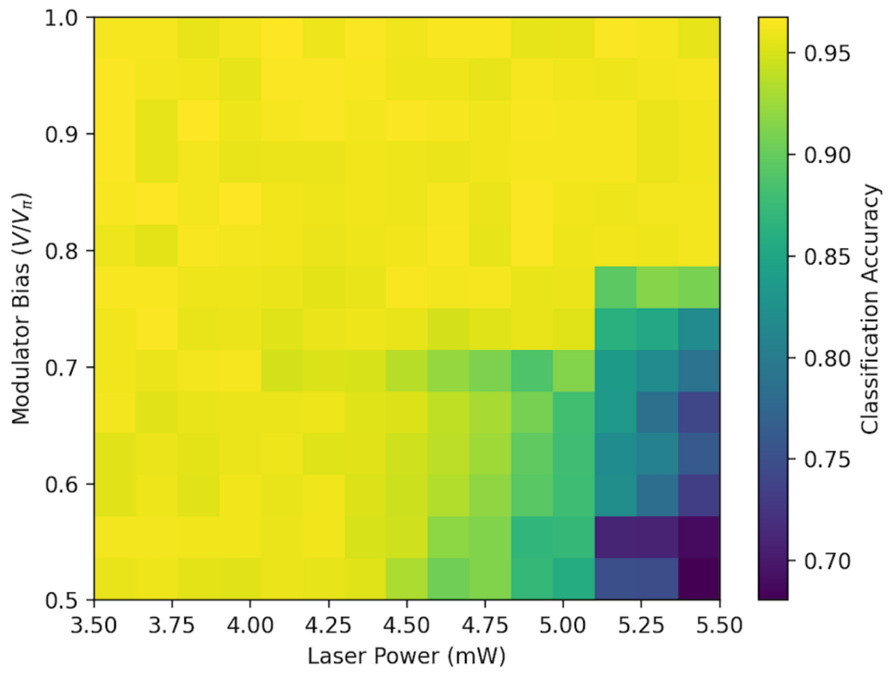
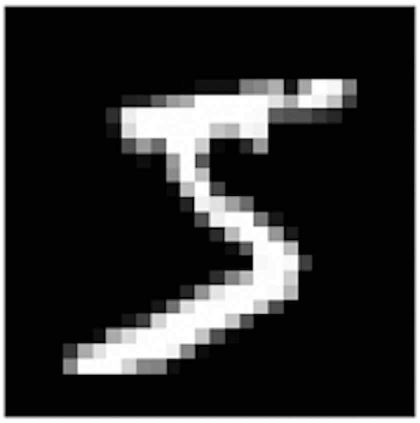


Fig. 2.16: Plot of experimental parameter sweep as a function of laser power and modulator bias. A modulator bias of 0 corresponds to the peak of the sinusoidal transfer function.

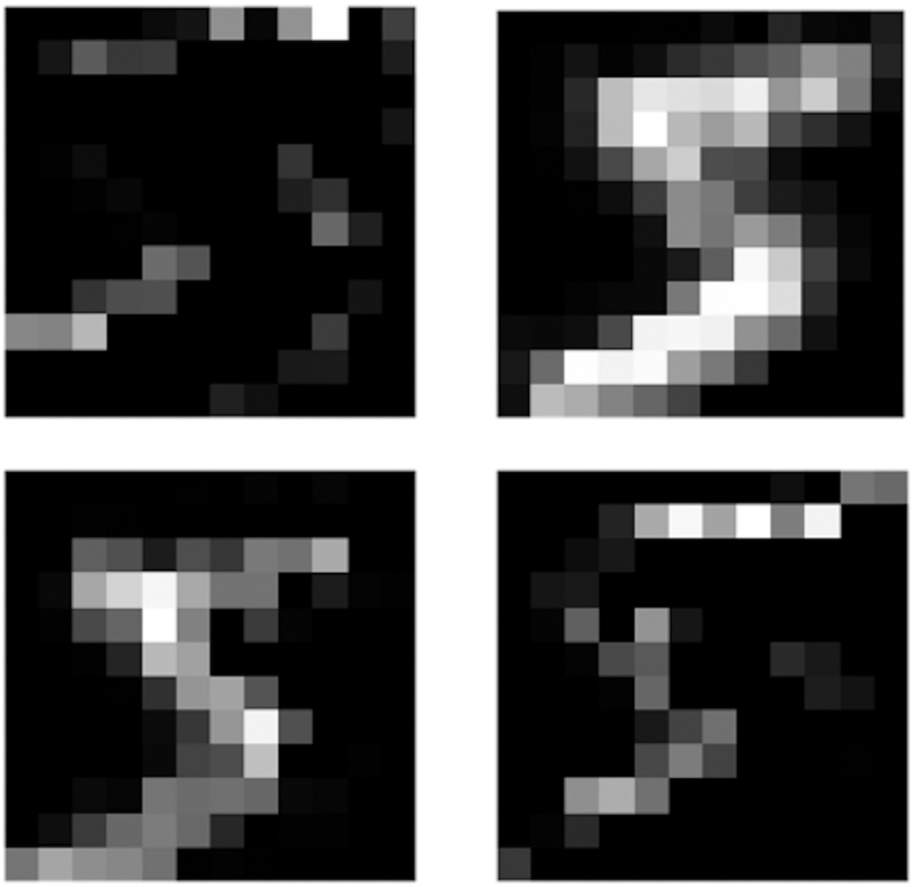
ranging in size from 1000 virtual nodes per image to 15000 virtual nodes per image, with results shown in Fig. 2.18. In simulation, we achieved test errors of 3.16% and 1.2%, respectively, at 1000 and 15000 virtual nodes, whereas we achieved 3.6% and 1.6% for for the same points in experiments. As benchmarks, we also plot performance of a standard RC scheme (i.e. no convolutional preprocessing) and a well-known CNN model, LeNet-5. For all reservoir sizes, our hybrid scheme, even with random convolutions, improves test accuracy by more than a percentage point in comparison to the standard implementation in both experiment and theory. At 15000 virtual nodes, we find our hybrid RC scheme comes close to the 0.95% test accuracy achieved using LeNet-5 [53]. Additionally, simulations of our scheme outperform the ELM baseline, indicating the added benefit of including RC memory.

2.4.3 Trained Hybrid RC

As an alternative approach, we also consider ways of implementing our hybrid RC scheme using trained weights in the convolutional layers, rather than simply randomly initializing them. The most straightforward way of achieving this is to fully train a CNN using backpropagation, after which we take the trained convolutional layer weights for use in our preprocessing procedure. For this purpose, we use a 6-layer CNN, denoted MNIST convnet in Table 2.5, containing the aforementioned 2 convolutional layers and one max pooling layer, followed by two fully-connected layers with 128 and 10 neurons, respectively for classification [17]. In total, this CNN model contains 1.2 million trainable parameters, with around 18000 of them contained within the two convolutional layers. Training this



(a)



(b)

Fig. 2.17: (a) Example image from MNIST dataset of digit "5". (b) Several feature maps generated by forward pass through two untrained convolutional layers.

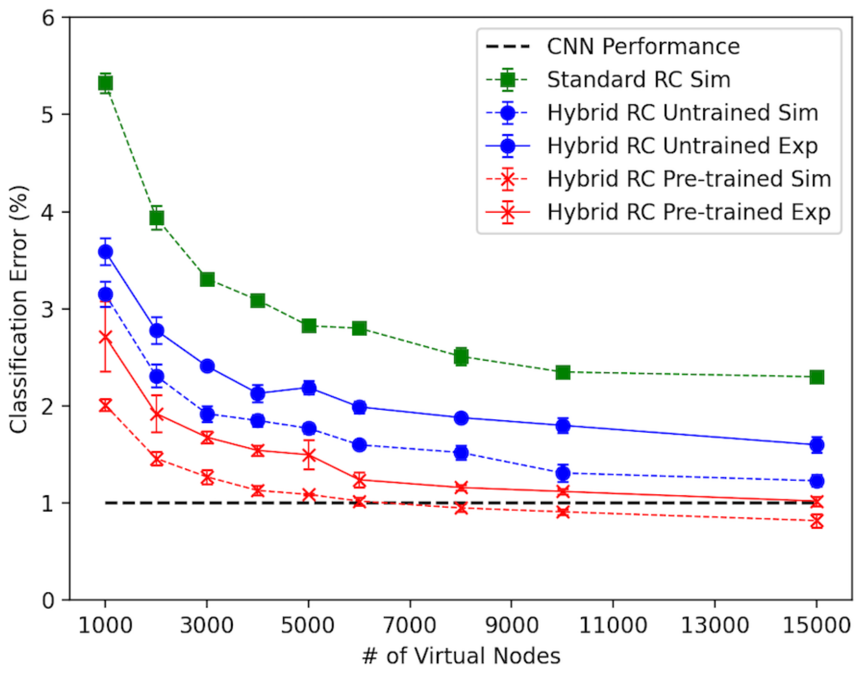


Fig. 2.18: Test error on MNIST task plotted as a function of virtual nodes for standard RC, hybrid untrained RC, and hybrid trained RC, both experiment and simulation.

model to convergence achieves around 1% test error, similar to that of LeNet-5. Using the two trained convolutional layers and max pooling layer for preprocessing, we replicate the experiments in Sec. 2.4.2, also plotted within Fig 2.18. In simulation, we achieve 2% and 0.8% test accuracy at 1000 and 15000 virtual nodes respectively, whereas we achieve 2.7% and 1.1% test accuracy in experiments. Particularly for smaller numbers of virtual nodes, pre-training the convolutional weights significantly improves test accuracy, as seen in the large divergence between the red and blue curves in Fig 2.18. We also note that at 15000 virtual nodes, our hybrid scheme in simulation actually outperforms the CNN model from which we have derived our trained weights from, as well as outperforming the LeNet-5 benchmark. Experiments nearly match these results, with test accuracy virtually indistinguishable from that of a CNN.

2.4.4 Experimental Comparison

In both the trained and untrained hybrid RC implementation, we are able to achieve test accuracy nearly equal to that of state-of-the-art CNN models for the MNIST task. Additionally, our experimental optoelectronic implementation offers further benefits compared to most digital machine learning algorithms. Most salient of these is the potential for significant increases in processing speed. The total processing time for our untrained hybrid RC implementation can be broken down into four parts: the convolutional preprocessing, the mask matrix multiplication, the reservoir circulation, and the final linear regression. Even for large reservoirs of 15000 nodes or more, the theoretical minimum processing time through the reservoir computer is quite small, requiring only a few seconds using a

Method	Test Error	Training Time*
Le-Net 5 CNN [53]	0.95%	1
MNIST Convnet	1.0%	1
Hybrid RC Untrained (Simulation)	**1.2%**	15
Hybrid RC Untrained (Experiment)	**1.6%**	**0.1**
Hybrid RC Trained (Simulation)	**0.8%**	15
Hybrid RC Trained (Experiment)	**1.1%**	**1.1**
Standard RC	2.3%	0.1
ELM	1.4%	0.25
Linear Classifier [53]	7.6%	0.01

Table 2.5: Comparison of hybrid RC approach with various other machine learning methods

*Normalized to LeNet-5 CNN case

125 MHz FPGA clock. To provide an estimate of speed increase, we compare the processing time of these four steps using our laptop's CPU, versus fully training a LeNet-5 network to convergence on the same CPU. Comparing these, our hybrid scheme can offer a potential 10× increase in processing speed compared to the training of the CNN, with the processing speed bottleneck arising from the large matrix multiplication required to produced the masked inputs. The full breakdown of our calculated hybrid RC processing time is as follows: 15% is spent performing convolutional prepocessing, 55% is spent performing the random mask matrix multiplication, 10% is spent circulating the virtual nodes through the reservoir, and 20% is spent running the ridge regression optimization. In total, the CNN training time is 10 minutes, whereas our hybrid RC training routine can be condensed down into as short as approximately one minute.

The trained hybrid RC scheme does not offer this same speed advantage, given that it requires the training of a CNN to select the desired weights. Thus, the trade-off between these two approaches is between a very fast solution with slightly worse performance, versus a slower solution, but with performance even surpassing many state-of-the-art algorithms. These results comparing processing speed and performance are summarized in Table 2.5. A potential middle ground exists in the leveraging of transfer learning, in which convolutional layer weights are taken from an already on-hand CNN model trained for a different task, from which some feature extraction information is still preserved [76].

2.4.5 CIFAR-10 Experiments

In addition to our experiments using the MNIST dataset, we also consider a much more challenging problem setting: CIFAR-10 image classification. CIFAR-10 is a dataset of 60000 32x32 RGB images consisting of 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks [43]. Given the larger diversity of object types and the inclusion of multiple color channels, experiments on CIFAR-10 are a good gauge of how our form of hybrid RC can scale to tackle more challenging problems.

Our experimental setup for CIFAR-10 experiments is largely the same as previously outlined, except we swap the CNN architecture used in the convolutional preprocessing stage with that of a VGG-8 network. VGG is an improved convolutional backbone over that of LeNet-5, capable of extending layers depths up to 19 layers with increasing performance [96]. A diagram of the VGG-8 architecture is included in Fig. 2.19. Training the VGG-8 network on CIFAR-10 classification achieves around 80% test accuracy without any significant bells and whistles. We perform RC experiments using both untrained and trained VGG-8 convolutional layers.

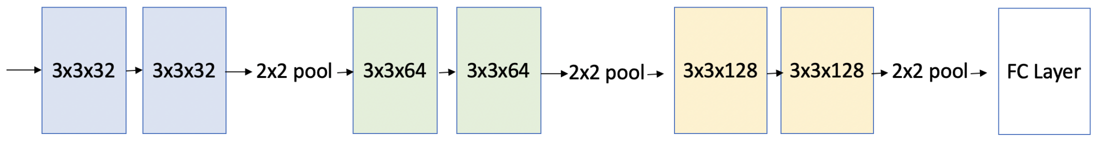


Fig. 2.19: Diagram of VGG-8 network architecture, with 6 convolutional layers, 3 pooling layers, and 2 fully-connected layers.

We plot the results of our CIFAR-10 experiments in Fig. 2.20. In contrast to the MNIST task, the performance of our untrained hybrid RC lags significantly behind the CNN baseline, with a gap of more than 25% in test accuracy at 10000 virtual nodes. Using pretrained weights in the convolutional preprocessing stage closes this gap significantly, but the test accuracy of our approach is still around 5% worse using 10000 virtual nodes. These results reveal the limitations of our hybrid RC approach; the feature extraction power of untrained convolutional layers are limited and not sufficient for more challenging image classification tasks. Using trained weights in the convolutional layers can bring hybrid RC performance close to that of standard CNNs, however the cost of training these layers becomes more significant. Ideas from transfer learning remain a potential direction for developing more advanced hybrid RC systems.

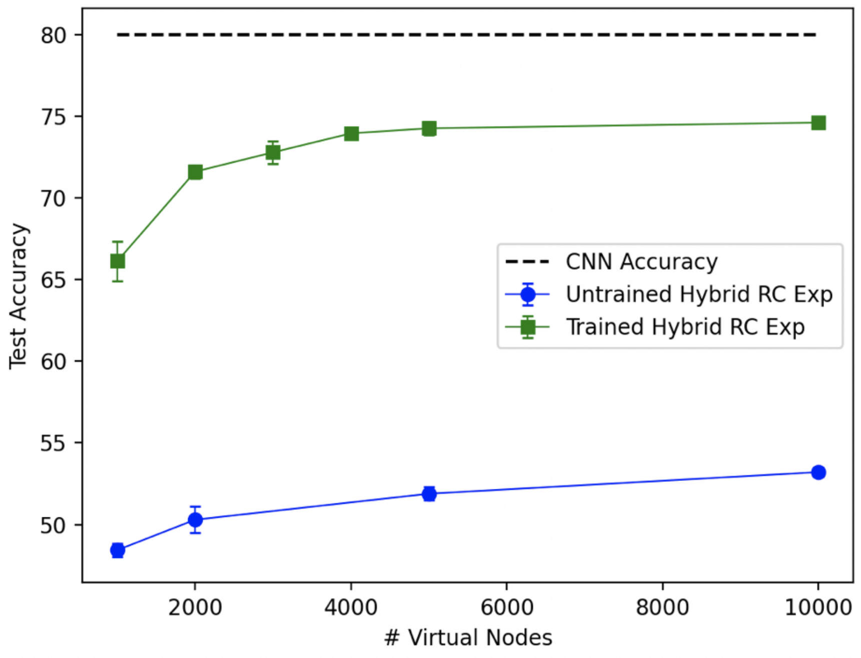


Fig. 2.20: Test accuracy on CIFAR-10 task plotted as a function of virtual nodes for hybrid untrained RC and hybrid trained RC experiments with VGG-8 performance for comparison.

2.5 Discussion

In this chapter, we introduced a new form of reservoir computing which combines convolutional preprocessing with an optoelectronic delay-based RC implementation. Our experimental platform combines digital pre- and postprocessing on a laptop CPU with a custom-programmed FPGA for interfacing with the analog optical/electronic reservoir computer. The main strength of this approach is a significant reduction in training time (even with limited hardware). On the MNIST handwritten digit recognition task, our approach achieves test accuracy on-par with a standard CNN approach, while requiring as little as a tenth of the time for training.

Although our approach achieves impressive results on the MNIST task, the technology still has significant limitations, making any deployment in an autonomous driving setting unlikely in the near future. Although capable on the simple MNIST dataset, autonomous driving tasks require perception systems far more discerning for challenging tasks such as vehicle, pedestrian, sign, and road detection and classification in the wild. Scaling our hybrid RC system to tackle problems on this scale is challenging; our experiments on the more-challenging CIFAR-10 classification task show that RC still lags significantly behind even simple CNN models on this more-representative dataset. In addition, MNIST and CIFAR-10 contain extremely low-resolution images (28x28), far below what typical HD cameras produce in modern vision systems. How to scale our form of delay-based RC to handle a greater number of input features while still maintaining a considerable latency advantage requires significantly more thought. Lastly, our current RC implementation, while potentially offering a significant training-time advantage, does not improve model latency in inference. For more niche AV applications requiring on-board training in real-time, this may be suitable. However, a lack of inference acceleration limits the desirability of such a system in production.

While significant hurdles remain in bringing reservoir computing to AVs, this work nonetheless represents one of the first explorations of applying RC to the domain of computer vision. Our results open a new path to achieving faster online training of edge vision systems. Future research directions from this work include exploring hybrid RC for vision with explicit temporal dependence (i.e. video) and leveraging techniques such as transfer learning to test the limits of combining random computations with pretrained models for faster training.

Chapter 3

Efficient LiDAR-Camera Fusion

The previous chapter of this thesis addresses the training efficiency of vision algorithms from a combined hardware/algorithms perspective. While reservoir computing shows some promise, the technology is not yet capable of meeting the needs of current AV perception. Thus, moving forward we take advantage of mature GPU hardware and instead focus on improvements from a purely algorithmic/data perspective. In this chapter we will address improving the speed and accuracy of algorithms for object detection in self-driving scenarios. Specifically, we introduce a new approach for LiDAR-camera fusion, which we dub Center Feature Fusion (CFF) which performs feature-efficient sensor fusion while achieving good performance on the nuScenes autonomous driving dataset. This chapter is based on work previously published in [35].

3.1 3D Object Detection Overview

3D object detection is a central pillar of the perception system for modern autonomous vehicles; being able to detect relevant objects (e.g. cars, pedestrians, cyclists) in a driving scene is key to ensuring safe navigation. Algorithms for 3D object detection differ depending on the sensor suite used on the vehicle, encompassing camera-based, LiDAR-based, radar-based, or some combination of these.

LiDAR is the main workhorse in most AV detection systems, providing low-resolution but accurate spatial and depth information about the scene. Cameras are ubiquitous in AV perception and beyond due to the technology maturity, providing high resolution color and texture information regarding the scene. Radar offers a low-cost alternative to LiDAR, providing 3D sensing even in poor conditions, although at a much lower resolution in comparison to LiDAR. Sensor fusion is the process of extracting and combining useful complementary information from each of these sensor modalities.

3.1.1 LiDAR-based 3D Object Detection

LiDAR-based 3D object detectors aim to detect objects with a given point cloud, represented as an unordered set of 3D points in space. In contrast to image-based deep learning, point cloud-based detectors lack a canonical backbone architecture due to the challenge of dealing with the irregularity of point clouds. Instead, different strategies exist depending on the application. A comparison of these is shown in Fig. 3.1.

Deep learning on indoor point clouds, which typically are smaller in number of points (on the order of thousands) and have more regular point density, generally employs a form of bottom-up grouping to enable hierarchical learning. The first works to explore this strategy, PointNet and PointNet++ [83, 84], operate directly on the point sets with an MLP (multi-layer perceptron) and group points using either K nearest neighbor (kNN) or a ball query. DGCNN builds edge graphs between points and processes them with graph convolutional layers [115]. VoteNet generates learned "vote" points and aggregates point cloud points around these based on Euclidean distance [81]. Bottom-up grouping works well as it allows for a finer-grained grouping of points accounting for differing point geometries and densities throughout the point cloud. However, the techniques used for these grouping strategies (e.g. farthest point sampling) are computationally slow and incompatible with parallel processing, making processing of large-scale point clouds computationally inefficient.

Deep learning on outdoor LiDAR point clouds (as is typical for autonomous driving perception) has to account for a larger number of points (on the order of hundreds of thousands or even millions of points) and very high sparsity. In this setting, the standard approach is to employ a form of top-down grouping, i.e. voxelization, to efficiently group points in 3D space. Voxel-based methods first quantize point clouds into regular 3D voxels before the voxelized representation is processed through the detector stage. VoxelNet [139] discretizes the 3D space before processing points inside each voxel with a PointNet-like network; voxels are then processed using a dense 3D convolutional neural network. SECOND [123] improves on VoxelNet by introducing efficient sparse convolutions. Pillar-based methods discretize the point cloud into vertical columns, flattening the representation along the height dimension. PIXOR [124] flattens the point cloud into a 2D pseudo-image which is encoded with a 2D CNN. PointPillars [48], similar to VoxelNet, processes points inside pillars through a PointNet before the encoded pillars are passed through a 2D convolutional network. PV-RCNN employs a hybrid point-based and voxel-based approach for detection [94]. CenterPoint replaced the standard two-stage detection architecture with an anchor-free approach by utilizing a center-based representation of detected objects, achieving the new state-of-the-art on 3D detection tasks [127].

3.1.2 Camera-based 3D Object Detection

3D object detection using only RGB camera images, although attractive as an alternative to expensive 3D sensors, still lags significantly behind LiDAR-based detection. Single-view 3D object detection in particularly is challenging due to the problem of monocular depth estimation. Some methods use standard 2D CNN backbones and attempt to directly estimate depth from single images to calculate 3D bounding boxes [138, 74]. Pseudo-lidar methods use a pre-trained depth estimation network to transform RGB images into point clouds that are then compatible with standard LiDAR-based detectors [114, 86]. Another class of detectors transforms camera features into a grid-based birds-eye-view (BEV) representation by predicting a per-pixel depth distribution [88, 60].

In contrast to single-view 3D detection, computer stereo vision seeks to estimate depth information by using images of the same scene taken from differing vantage points. In the context of autonomous vehicles, many works seek to leverage vehicle motion to generate differing views of a scene from a single camera view [113, 59]. Although stereo vision improves depth estimation over single-view detection, stereo vision fails for objects lacking significant variation in features or textures.

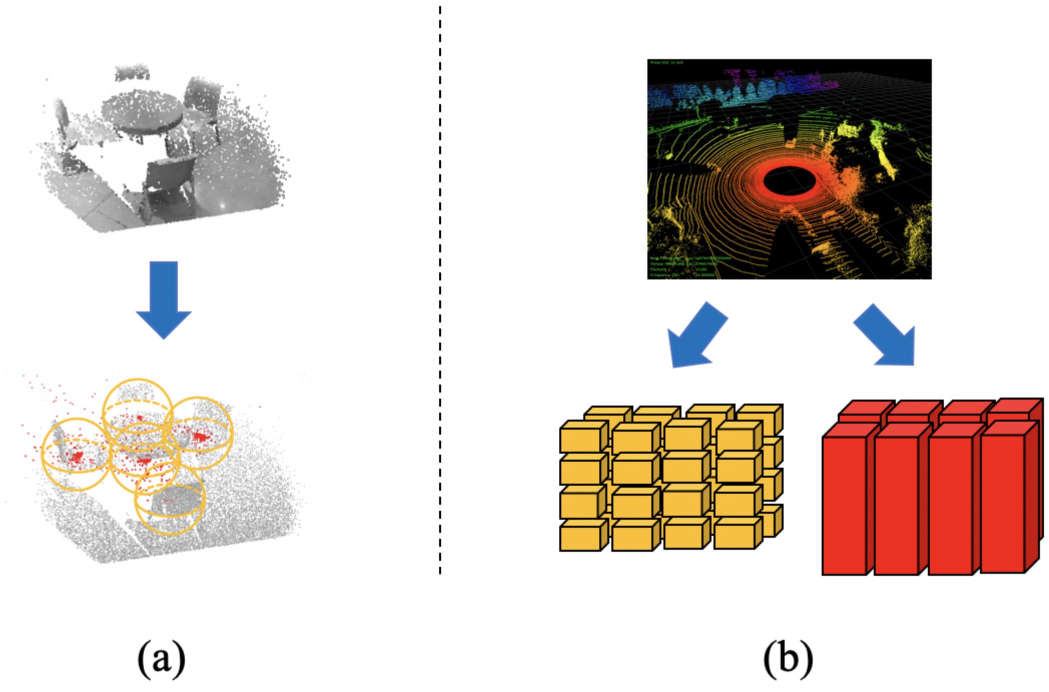


Fig. 3.1: (a) Bottom-up grouping used in indoor point clouds. (b) Top-down grouping (i.e. voxelization) employed for outdoor point clouds.

3.1.3 Multi-modal Fusion

Sensor fusion, specifically LiDAR-camera fusion, seeks to leverage the relative strengths of both sensor modalities and combine them to most meaningfully improve the detection pipeline. Generally, fusion approaches keep most of the detector backbones of each respective modality intact, while introducing information exchange at various point(s) in the pipeline. These can be broken down into three broad strategies: early fusion, middle fusion, and late fusion. Early fusion seeks to fuse information at the input level (i.e. at the raw pixel/point level). Middle fusion performs fusion at the deep feature level after raw inputs have been encoded with a neural network. Late fusion fuses the outputs of the model (bounding boxes in the case of object detection). A schematic comparing these approaches is shown in Fig. 3.2.

Early sensor fusion work focused primarily on proposal-level fusion, i.e. fusing information at the region proposal level (a form of mid fusion). MV3D [12] fuses object proposals in three views (BEV, front view, and perspective view), whereas AVOD fuses them in two views (BEV and perspective view) [46]. RoarNet determines geometrically feasible 3D poses from images to generate a set of 3D region proposals [95]. Frustum-PointNet [82] and Frustum-ConvNet [116] lift image proposals into 3D frustums to identify relevant regions of the point cloud. Later works such as FUTR3D [13] and TransFusion [6] generate object proposals in 3D and refine them using a transformer decoder which attends to 2D features. By nature of fusing information at an object proposal level, these fusion approaches are specific to the task of object detection and cannot be generalized to other tasks.

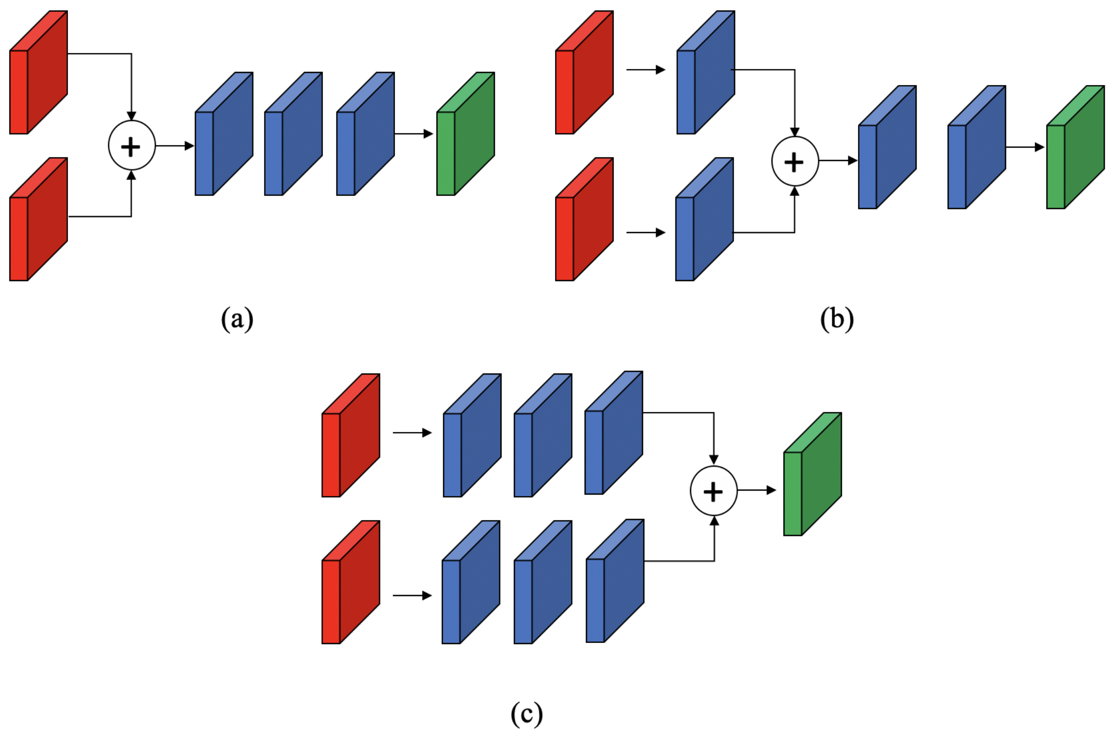


Fig. 3.2: High level comparison of (a) early (b) middle and (c) late fusion strategies.

Point decoration, a hybrid form of early fusion, has become a popular paradigm in sensor fusion for the past few years. The general outline of point/input decoration is as follows: first, camera images are encoded with a neural network to generate deep features in the image space. Next, the LiDAR data (points or another representation) are projected to the camera space, where each point is associated with a corresponding pixel. Then, the corresponding camera features are appended to the points, before being processed through a standard LiDAR-based backbone for detection. A visualization of point decoration is shown in Fig. 3.3. PointPainting [110] proposed projecting the point cloud onto the outputs of 2D semantic segmentation networks to augment each point with the class label of the corresponding 2D pixel. FusionPainting [122] fuses segmentation labels from both 2D and 3D segmentations to generate more accurate labels for the point cloud. PointAugmenting [111] instead annotates points with deep CNN features while processing the LiDAR and camera features through separate streams in the CNN backbone. AutoAlign [15] uses a learnable cross-attention module to associate pixel-level features with voxels, instead of the hard association of projection using a camera matrix. AutoAlignV2 adopts a more efficient approach that first projects voxel features to the camera coordinate system, followed by a deformable cross-attention operation to associate pixel features with voxels. MVP [128] takes a different input decorating approach, projecting the point cloud onto a segmentation map to generate virtual LiDAR points inside each object to densify the point cloud. While variations of input decoration have dominated much of the recent research in sensor fusion, the approach suffers from several inherent drawbacks. Most significant of these is point decoration approaches only augment the existing LiDAR data with additional camera features, failing to leverage the higher resolution of camera images to compensate for low-resolution LiDAR point clouds.

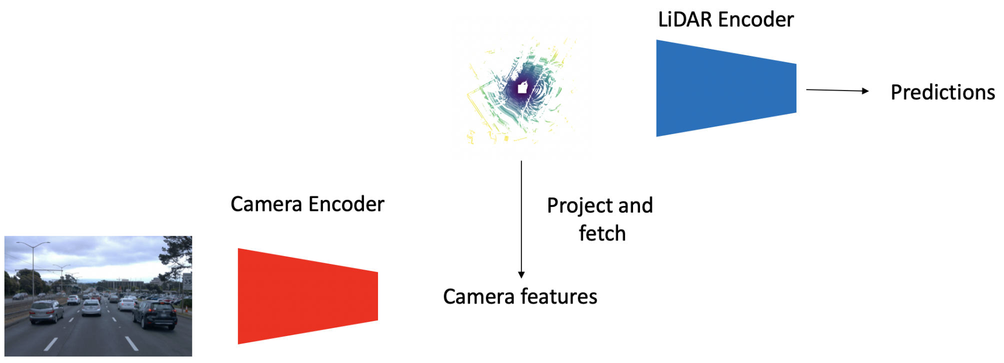


Fig. 3.3: Illustration of point decoration strategy for sensor fusion.

Concurrent to our work has been a trend toward mid-level deep feature fusion. In contrast to point decoration, deep feature fusion is a form of mid fusion in which deep features from both the camera and LiDAR streams are fused together. Fig. 3.4 displays a visualization of this fusion scheme. Continuous Fusion [61] shares information between the 2D and 3D backbones at all layers of the neural network. DeepFusion [58] uses a learnable attention layer to associate CNN features between different sensor modalities. Two works concurrent to ours, both dubbed BEVFusion [67, 105], use the approach known as lift-splat-shoot (LSS) [80] to generate per-pixel depth predictions to project the full set of camera features to BEV, where after undergoing a pooling operation, they are concatenated with LiDAR BEV features. While fusion in BEV space is a promising approach we will discuss in more detail later in this chapter, BEVFusion is limited by both the latency associated with projecting and pooling the camera features, and the inherent accuracy in depth estimation with an approach such as LSS.

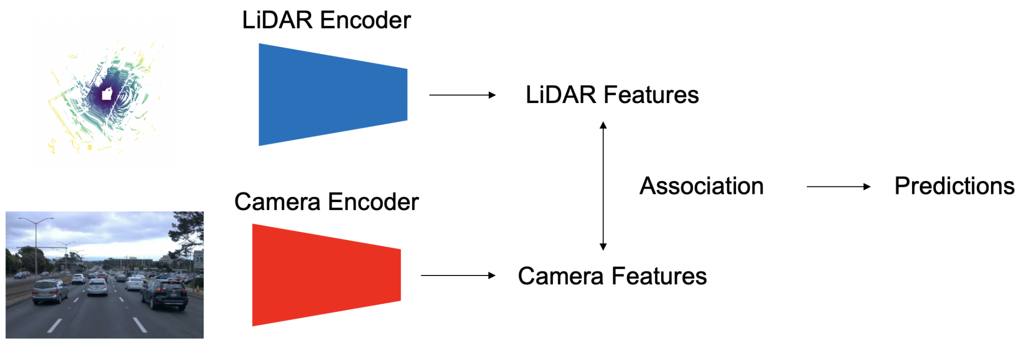


Fig. 3.4: Illustration of deep feature fusion strategy for sensor fusion.

3.2 Center Feature Fusion

In this section, we introduce a novel approach for fusion-based 3D object detection, which we dub Center Feature Fusion (CFF).

3.2.1 Overview

CFF approaches sensor fusion in a fundamentally different manner than previous input decoration approaches by instead considering fusion in BEV space. CFF adopts the approach of other mid fusion approaches and first encodes the LiDAR point cloud and camera images with LiDAR and camera backbones, respectively. We use a Centerpoint-based Voxelnet as our LiDAR backbone, and DLA-34 as our image backbone. The top-down bird's-eye-view is a naturally attractive space for sensor fusion as it preserves the geometry of the scene (in contrast to camera perspective view) while also serving as the space in which the planning algorithms of the autonomy stack operates. However, performing fusion in BEV space is not straightforward; transforming camera images from perspective view to BEV requires per-pixel depth measurements, which RGB cameras naturally lack the capability to capture. Concurrent approaches try to predict a probabilistic depth distribution over discrete bins, however this is both inherently inaccurate and costly in terms of latency. Instead, we leverage well-calibrated sensors and transform the point cloud to perspective view, followed by a depth interpolation to generate a dense per-pixel depth map. Using this depth map, we can accurately project camera images to BEV. However, with high resolution input images, this operation can be prohibitively costly from a latency perspective to perform. Instead, we leverage camera-based center-based detection (CenterNet) to identify the key pixel features, and instead project only these (a small fraction of the total) to BEV. These features are concatenated with the LiDAR BEV features, after which they are further encoded with a small number of convolutional layers before being passed to the detection heads. Lastly, data augmentation is a key component of training 3D object detectors. LiDAR-based data augmentation consist of geometrical perturbations to the point cloud (e.g. random rotation, translation); however applying these in a fusion context destroys alignment with camera projections. Instead, we modify the augmentation scheme to apply augmentations to the projected camera images in BEV, as opposed to the perspective view images, in order to preserve alignment between the two modalities. A detailed diagram illustrating CFF is shown in Fig. 3.5.

3.2.2 LiDAR Backbone

To process the LiDAR point cloud, we adapt a VoxelNet-based backbone. First, the point cloud is partitioned into 3 dimensional voxels of size (0.075 m, 0.075 m, 0.2 m). Using the standard point cloud range outlined by the nuScenes dataset ($\pm$54m in the x and y directions and -5m to 3m in the z direction) results in a voxelized representation of $1440 \times 1440 \times 40$. Dynamic voxelization is used for efficient memory storage. To generate a set of feature vectors for each voxel based on the points contained within, we use a "simple" encoding approach of averaging the point features of all points contained within the voxel. Next, the encoded voxels are processed through a set of convolutional middle layers, implemented using sparse convolution tensor operations. The convolutional middle layers contain a set of 4 3D convolution blocks with channel numbers of 16, 32, 64, 128 respectively. After the convolutional middle layers, the features are passed through an additional two convolution blocks, increasing the channel number to 256. The output feature map is reduced in the xy direction by a factor of 8, and is flattened along the z direction, comprising a solely top-down view of the scene.

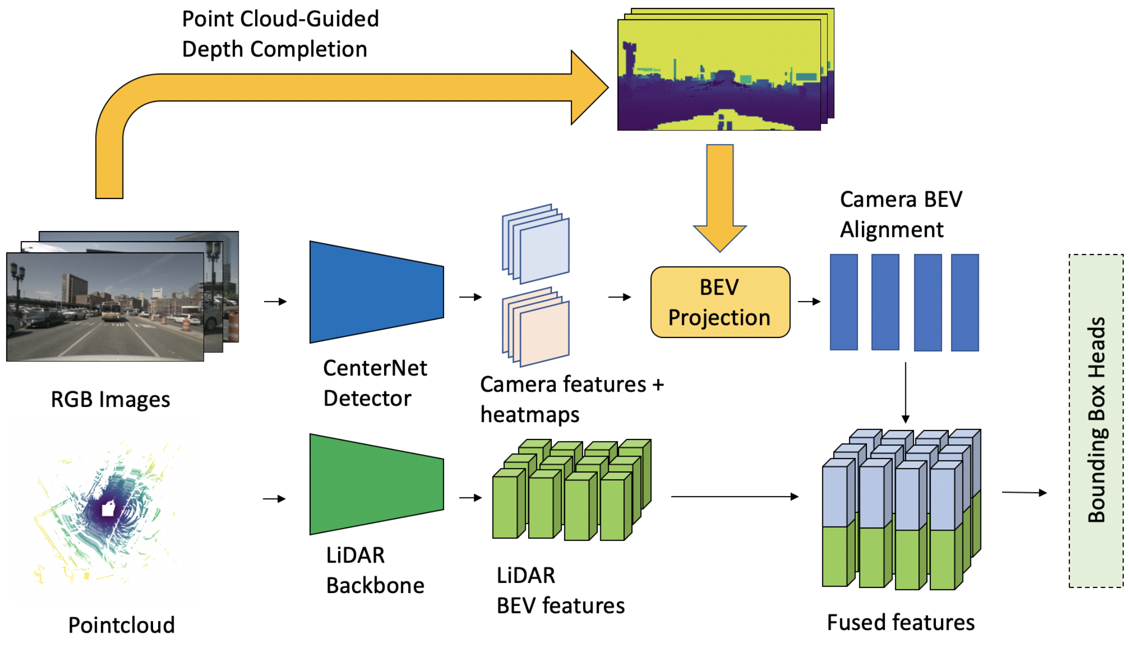


Fig. 3.5: Summary of our proposed Center Feature Fusion model. The point cloud and camera images are first encoded using a VoxelNet and DLA34-CenterNet, respectively. Our selective projection is performed on the camera features to project them to BEV where they are then fed into a small 2D CNN to help with feature alignment. The multi-modal BEV features are then concatenated and passed into CenterPoint's bounding box heads.

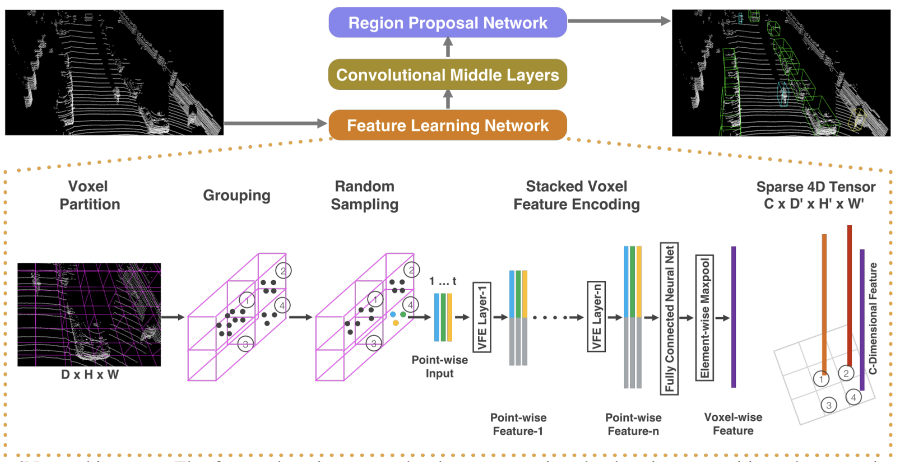


Fig. 3.6: Overview of VoxelNet architecture. Figure adapted from [139].

3.2.3 Camera Backbone

To process the camera images, we adapt the same backbone used in [138], DLA-34. DLA (deep layer aggregation) is a modern CNN backbone network which hierarchically aggregates feature maps to better propagate features through the network. Instead of a standard 3×3 convolution operation, our DLA-34 backbone uses deformable convolutions in its convolution blocks [19]. The output of the DLA-34 backbone generates feature maps with 256 channels and a height and width reduced by a factor of 4 from the input.

3.2.4 Point Cloud-Guided Depth Generation

After encoding both camera images and the LiDAR point cloud with their respective backbones, the next step is to perform feature fusion. As we propose fusing the two sets of features in BEV, the first step is completing the required transformation of the encoded camera features. The transformation from camera perspective view to BEV requires per-pixel depth estimates, which are naturally lacking from the camera features. However, operating under the assumption that the camera and LiDAR sensors are well callibrated, we have the ability to transform the point cloud to the camera frame(s) via a homogenous transformation. To do so, we consider two quantities which define the camera coordinate system: the extrinsic and intrinsic matrix. The extrinsic matrix defines the translation from the world coordinate frame to the 3D coordinate system of the camera, defined by its pose (location and direction of view). The extrinsic matrix is composed of two transformations: a 3D rotation, and a 3D translation. The composed extrinsic matrix takes the following form:

$$E = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline 0 & 1 \end{array}\right) \tag{3.1}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector. The intrinsic matrix defines the perspective projection which transforms the 3D camera coordinate system to the 2D imaging plane coordinate system. The intrinsic parameters are defined by internal characteristics of the camera, taking the following form:

$$I = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

where f_x and f_y are the focal lengths in the x and y direction, respectively, x_0 and y_0 is the principal offset, and s is the axis skew. The full transformation from 3D world coordinates to the 2D imaging coordinates is defined as a composition of the intrinsic and extrinsic matrix:

$$P = I \times E \tag{3.3}$$

After transforming the point cloud to the camera frames corresponding to each directional-facing camera on the vehicle, points outside of the camera field of view are cropped, leaving only the points which occupy the same coordinate as an existing pixel on the imaging plane. For pixels matched to these points, we can directly infer their depths from the z coordinate of the corresponding point in the 3D camera coordinate frame. However, because the LiDAR imaging resolution is significantly lower than that of the camera, the majority of pixels will not have a matched point to use for determining its depth value. Thus, we are required to perform depth completion, the task of converting a sparse depth map to a dense one. While several deep learning-based approaches exist, they both require a separate pre-training phase and are computationally expensive [38, 106, 23, 121]. Instead, we leverage an approach known as IP-basic, a fast, lightweight and efficient classical depth completion algorithm which is significantly more accurate than a simple nearest-neighbor classifier [45]. IP-basic operates in the manner of a modified version of a nearest neighbor classifier, instead using custom kernels to fill in empty depth values in the image in a strategic way. Requiring no training data, we can use IP-basic at runtime with minimal added overhead. We use an interpolation-only version of the IP-basic algorithm (i.e. only predicting depths for pixels within the convex hull of the projected points) so as to ignore objects in the image beyond the LiDAR range. For pixels outside the interpolation range, we assign them large depth values outside of the detection range so that they remain ignored after projection to BEV. A summary of the entire depth generation process is visualized in Fig. 3.7.

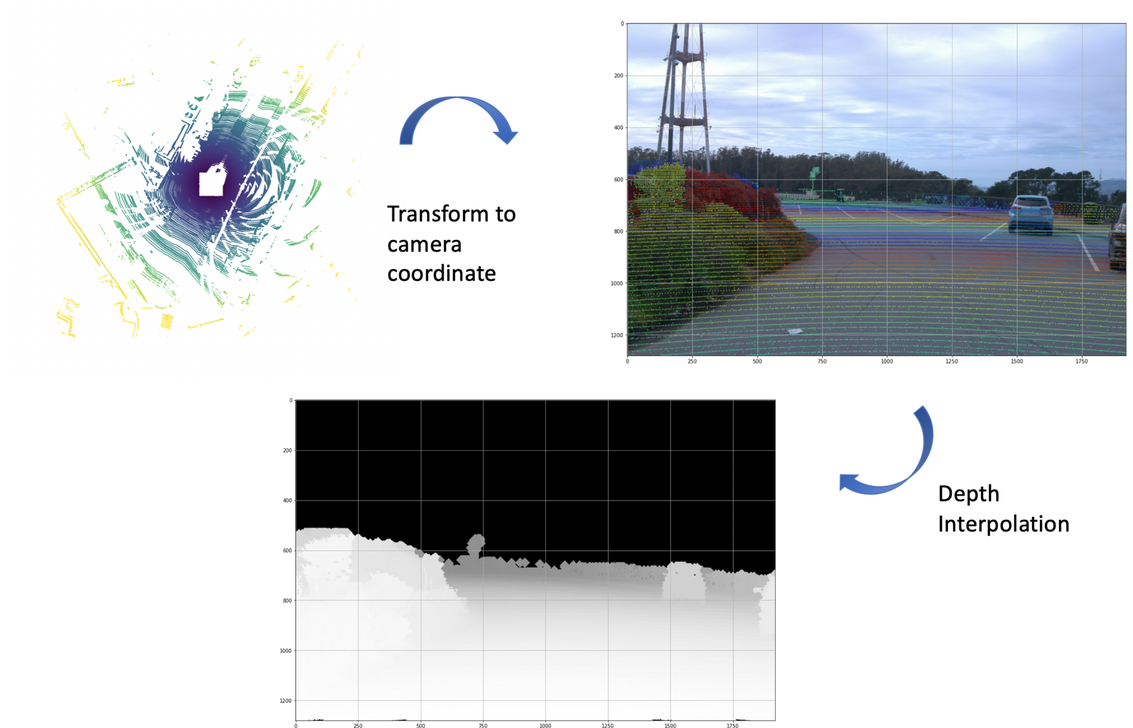


Fig. 3.7: Overview of point cloud-guided depth completion. First, the LiDAR point cloud is projected onto the camera's imaging coordinate system. An overlay of LiDAR point with an RGB image is shown to illustrate the alignment. Then, IP-basic is used to perform depth interpolation, generating a dense per-pixel depth map.

3.2.5 Selective Bird's-eye-view Projection

After generating a dense depth map for each set of camera feature maps, we now have the capability of projecting the camera features to BEV. Mathematically, this is achieved by simply performing the inverse of the camera transformation described in the previous section. Consider a camera pixel coordinate (u, v, d), where d is the predicted depth value for the corresponding pixel. The inverse of the perspective projection (i.e. transformation to the camera 3D coordinate system (x, y, z)) is given by:

$$x = d(u - x_0)/f_x \tag{3.4}$$

$$y = d(v - y_0)/f_y \tag{3.5}$$

$$z = d \tag{3.6}$$

We note that the camera intrinsics for the datasets we use in this work have axis skew $s = 0$, allowing us to utilize the above set of equations. To transform the camera features to the corresponding world coordinates, we simply invert the camera extrinsic matrix to apply the inverse affine transformation. To populate our BEV feature grid with camera features, we now simply take the camera features in world coordinates and sort them into discrete bins based on the predetermined resolution of the BEV feature map. Then, we perform max pooling on all of the features within a single BEV grid bin to generate the final BEV representation of the camera features.

While the described operation for BEV transformation is straightforward to perform, we note that the time complexity of the operation becomes untenable for a significant number of pixels. Even for a relatively modest resolution of 1600×900 pixels used by the cameras in the nuScenes dataset, a feature map downsampling factor of 8 combined with 6 directional cameras on the vehicle results in a total of over a million pixel features required to be projected, a heavy computational load. To circumvent this issue, we consider *selective projection*, where we aggressively filter the camera pixels *before* performing the BEV transformation, significantly reducing the runtime latency. In considering this approach, we note that for the task of object detection, only the camera features corresponding to objects of interest are relevant for fusion. As the vast majority of camera pixels correspond to background parts of the image (e.g. roads, buildings, etc.), only a small fraction of pixels contain the information most relevant to fuse.

To identify these key pixels, we leverage CenterNet, an efficient one-stage 2D object detector. CenterNet, in contrast to anchor-based object detectors, uses a detection head comprised of three sub-modules: a keypoint heatmap head, an offset head, and a size head. The keypoint head generates a heatmap $\hat{Y} \in [0, 1]^{\frac{w}{s} \times \frac{h}{s} \times K}$, where w and h are the input height and width, respectively, s is the downsampling factor of the output, and K is the number of classes. In the CenterNet model, objects are characterized by their center pixels, located in the keypoint heatmap through extracting all local maxima above some threshold value. Thus, higher heatmap values correspond to a higher "objectness" score for a particular pixel. Bounding box characteristics are then determined through the values predicted by the other two detection heads located at the corresponding heatmap peaks. Because CenterNet infers the bounding box characteristics from the feature located at the object center, we argue that its learned features contain all of the relevant object information condensed into a few select points. Fig. 3.8 contains a visualization of CenterNet's detection strategy.

We share the DLA-34 image backbone used for encoding the camera images with the CenterNet detector, and generate a center heatmap from the encoded features using

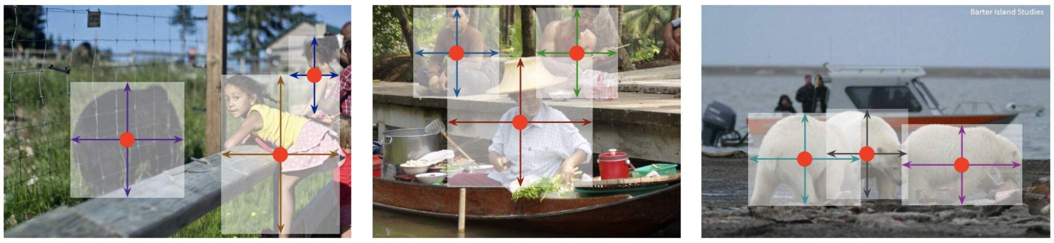


Fig. 3.8: Visualization of CenterNet's strategy of inferring object bounding box characteristics through a single feature corresponding to the object's center. Figure adapted from [138]

the aforementioned detection heads. To determine the pixels to project, we introduce a heatmap threshold hyperparameter; pixels with heatmap values above the threshold meet the importance requirement to be projected, whereas pixels below the threshold are disregarded. The features are fetched from the output of the DLA-34 backbone (i.e. we use the features before they are encoded by the detection heads). In total, 64 output feature channels are contained in the projected camera features. A lower threshold value, while more comprehensive in the feature fusion, increases runtime latency, whereas a high threshold value results in the opposite. Ablations of the heatmap threshold will be discussed in the experiments section of this chapter. Fig 3.9 illustrates the described selective projection routine.

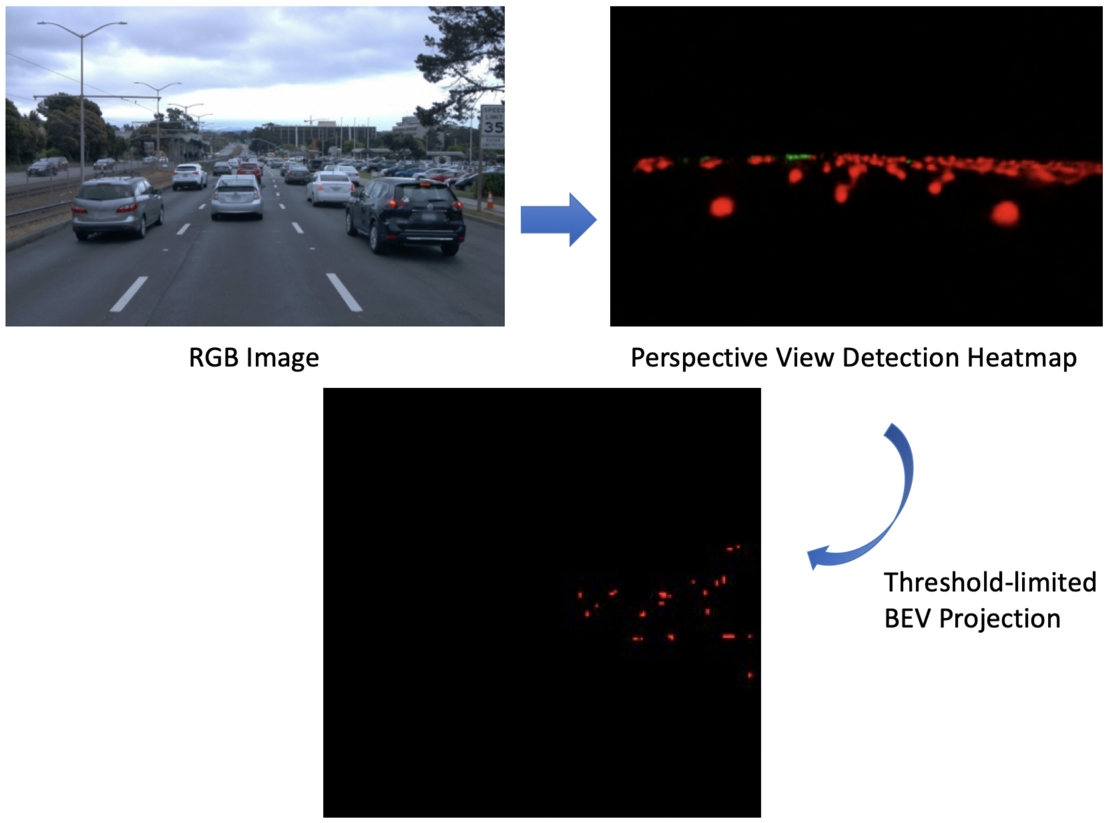

RGB Image
Perspective View Detection Heatmap
Threshold-limited BEV Projection

Fig. 3.9: Figure demonstrating center-selective BEV projection. First, CenterNet is used to generate a center detection heatmap. Based on the the predetermined threshold, camera features are then fetched and back-projected into BEV.

3.2.6 3D Detection Heads

After feature fusion is performed, the BEV feature map consists of a concatenation of both LiDAR BEV features from the VoxelNet backbone, and camera BEV features extracted from the DLA-34 backbone. To ensure strong fusion performance, accurate alignment between the features' of the two modalities is essential. However, errors arising from depth estimation and inaccurate sensor calibration introduce modest misalignments between the two sets of features in BEV. To correct for this, we additionally process the projected camera features through a series of eight 3×3 convolutional layers in order to increase the effective receptive field of each BEV feature, helping compensate for projection inaccuracies. We pad the feature maps to ensure that the output shares the same feature dimension as the original input.

After aligning the camera BEV features with the convolutional encoding, the fused features are encoded with a single share convolution before being passed to the detection heads. To perform the final detection, we adopt the same anchor-free detection procedure as is used by CenterPoint [127]. As is the case with CenterNet's 2D detection scheme, the 3D corollary uses a center heatmap head (in BEV) for object localization in conjunction with several regression heads: a sub-voxel refinement head (a 2D xy refinement), a height-above-ground head (to predict the z coordinate of the object), a 3D size head, a yaw rotation angle head (parameterized through predicting the sine and cosine of the rotation angle), and a 2D velocity head. As in CenterNet, object bounding boxes are extracted through finding local maxima in the BEV heatmap, and then fetching the corresponding box characteristics from the regression heads.

3.2.7 Model Training

3.2.7.1 Training Losses

To supervise our the training of our model, we use a loss function composed of two components: a classification loss, and a regression loss. The classification target is created in the standard manner for center-based detectors: ground truth objects are scattered onto the BEV grid target in a one-hot encoded manner. To increase the supervisory strength of the signal, Gaussians with peaks at each object center and radius $max((wl), 2)$ are drawn on the target heatmap. We use a focal loss summed over all elements of the BEV grid as the classification loss:

$$L_{cls} = \sum_{xy} \begin{cases} (1 - \hat{Y}_{xy})^{\alpha} \log(\hat{Y}_{xy}) & \text{if } Y_{xy} = 1 \\ (1 - \hat{Y}_{xy})^{\beta} \hat{(}Y_{xy})^{\alpha} \log(1 - \hat{Y}_{xy}) & \text{otherwise} \end{cases} \tag{3.7}$$

where $\hat{Y}_{xy}$ is the model prediction heatmap, Y_{xy} is the ground truth heatmap, and α and β are the focal loss hyperparameters (selected to be 2 and 4 respectively). To supervise the regression heads, we use an L1 loss evaluated at the center point of each object using the ground truth object parameters. The total loss is then:

$$L_{tot} = L_{cls} + \lambda L_{reg} \tag{3.8}$$

where we set $\lambda = 0.25$.

3.2.7.2 Training Hyperparameters

The first step in our training routine is to separately pretrain the image backbone. We use a pretrained CenterNet model due to the necessity of an accurate object heatmap for meaningful projection of camera features to BEV for fusion. Our pretrained CenterNet model is trained following the procedure outlined in [138]. While training the full fusion model, we freeze the weights contained within the image backbone. Similarly, we also load in the VoxelNet backbone with weights from a pretrained CenterPoint model to increase the speed of convergence. For training the fusion model, we use the adamW optimizer [69] with a one-cycle learning rate policy, a max learning rate of $1e-3$, weight decay of 0.1 and momentum between 0.85 and 0.95. We adopt the class-balanced sampling and class-grouped training proposed in CBGS, which re-samples training data samples to increase the frequency of rarer object classes during each training epoch [141]. We train our model using a batch size of 12 on 3 Nvidia A6000 GPUs for a total of 6 epochs.

3.2.7.3 Data Augmentation

Strong data augmentations are essential during training of 3D object detectors (particularly on small-scale publicly available datasets) to prevent overfitting and enhance the training data. In sensor fusion applications, data augmentation poses a challenge due to the fundamentally different approaches applied in 2D (e.g. random cropping, random flipping) and 3D (random rotation, scaling). Applying these augmentations naively destroys the consistency between fused points and degrades performance. DeepFusion addresses this problem with InverseAug, in which they save the 3D augmentation parameters, reverse them at the fusion stage, and then calculate the feature's corresponding coordinates in the 2D frame. By using BEV as our fusion representation, we are able to propose an even simpler algorithm we call ProjectionAlign. Because we project camera features to BEV, we generate a pseudo-3D point cloud, which can be augmented in the same manner as the LiDAR point cloud. Thus, we simply apply augmentations to the point cloud, save the exact parameters of the augmentation, and apply them to the camera features once they are converted to the same 3D world coordinate system. A visualization of the feature alignment achieved through ProjectionAlign is shown in Fig. 3.10. We perform the standard LiDAR data augmentations of random flipping along the X and Y axis (using 50% chance of flipping), global random scaling (scale value $\in [0.9, 1.1]$), global random rotation (angle in radians $\in [-\pi/4, \pi/4]$), and global random translation (gaussian distribution with $\sigma = 0.5m$). We don't apply any geometric augmentations to the camera images directly to preserve the exact alignment between the two modalities.

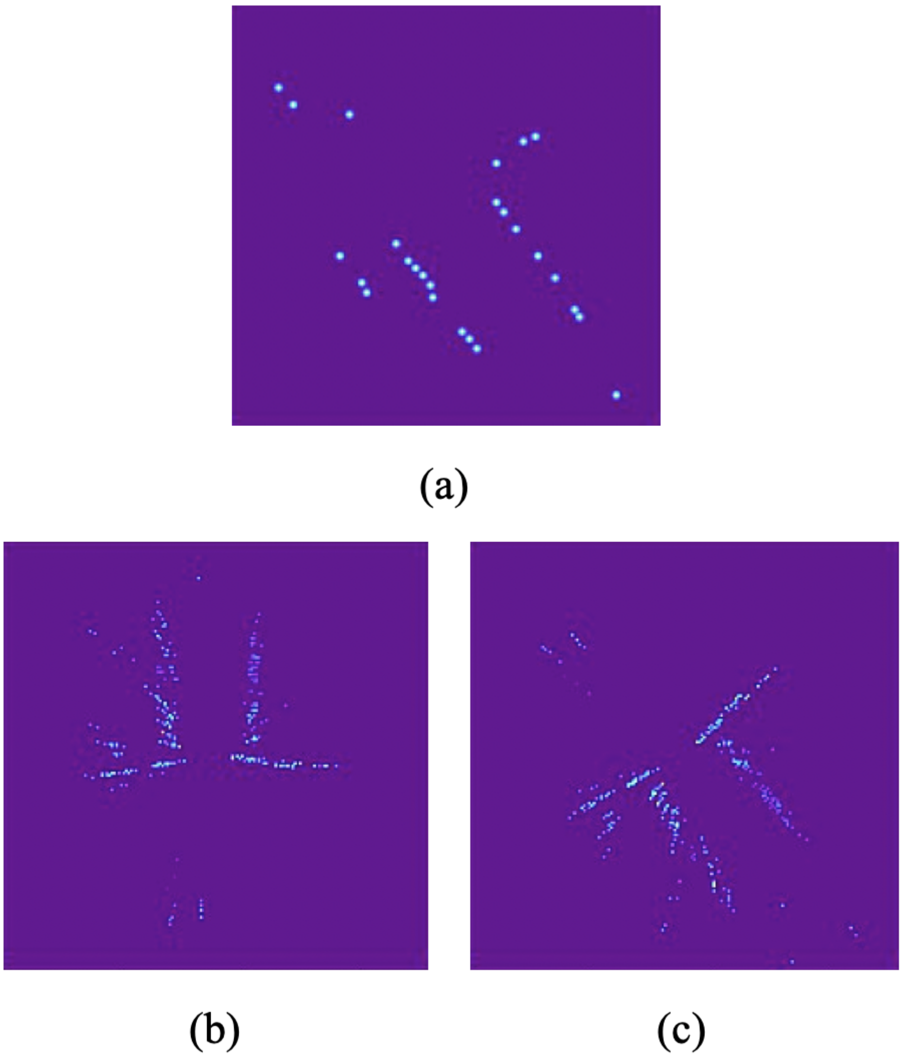


Fig. 3.10: Demonstration of ProjectionAlign. a) Ground truth detections in BEV, represented as points on the BEV grid. b) Unaligned projected camera features. c) Aligned camera features. Much stronger location correspondence exists between aligned features and ground truth object locations vs. unaligned features.

3.3 Experimental Results

3.3.1 Dataset

To evaluate CFF, we perform experiments using the nuScenes dataset, a large-scale autonomous driving dataset released by nuTonomy [10]. nuScenes consists of 1000 20 second driving scenes, split into 700 for training, 150 for validation, and 150 for testing. The scenes were collected in Boston and Singapore, both highly dense and challenging urban driving environments. Object annotations are included for every tenth frame in the dataset, resulting in 28130 frames for training, 6019 frames for validation, and 6008 frames for testing. While object labels are provided on the training and validation sets, test set labels are known only to the dataset curators and require submission to a public leaderboard for evaluation. As is standard for nuScenes, we form denser point clouds through concatenating points across all non-keyframe. Each frame consists of a 32-beam LiDAR scan and six monocular camera images of resolution 1600 x 900 covering the full 360-degree field-of-view. Annotations come from a set of 10 classes: cars, trucks, construction vehicles, buses, trailers, barriers, motorcycles, bicycles, pedestrians, and traffic cones. Sensor calibration parameters (e.g. camera intrinsics and extrinsics, LiDAR to world coordinate transformation) are included for each frame in the dataset. Ego car coordinates are calculated using a combination of GPS and IMU (inertial measurement unit) readings. HD semantic maps are also provided for each driving scene. Detection performance is measured using two main metrics: Mean Average Precision (mAP), and the nuScenes Detection Score (NDS). NDS is a custom metric consisting of an average of both mAP and five true positive metrics: mean translation error (mATE), mean size error (mASE), mean orientation error (mAOE), mean velocity error (mAVE), and mean attribute error (mAAE).

3.3.2 Main Results

First we compare the improvement from our proposed fusion approach against the CenterPoint LiDAR-only baseline. We display this comparison on the nuScenes validation set in Table 3.1. Our method achieves a +4.9% improvement in mAP and a +2.4% improvement in NDS overall against CenterPoint, with improvements in mAP seen across all of the classes. Fig 3.11 visualizes our results qualitatively for an example frame in the nuScenes validation set.

	mAP	NDS	Car	Truck	C.V.	Bus	Trailer	Barrier	Motor.	Bike	Ped.	T.C.
CenterPoint [127]	59.6	66.8	85.5	58.6	17.1	71.5	37.2	68.5	58.9	43.3	85.1	69.7
Ours	**64.5**	**69.2**	**85.5**	**59.1**	**22.5**	**73.4**	**43.2**	**68.8**	**72.8**	**62.1**	**86.4**	**70.6**
Improvement	+4.9	+2.4	+0.0	+0.5	+5.4	+1.9	+6.0	+0.3	+13.9	+18.8	+1.3	+0.9

Table 3.1: Performance comparison of CenterPoint and our method on nuScenes validation set. We report the total Mean Average Precision (mAP) and nuScenes Detection Score (NDS), and per-class mAP. C.V., Motor., Ped., and T.C. are short for Construction Vehicle, Motorcyle, Pedestrian, and Traffic Cone, respectively.

Table 3.2 compares the performance of our method with other state-of-the-art fusion

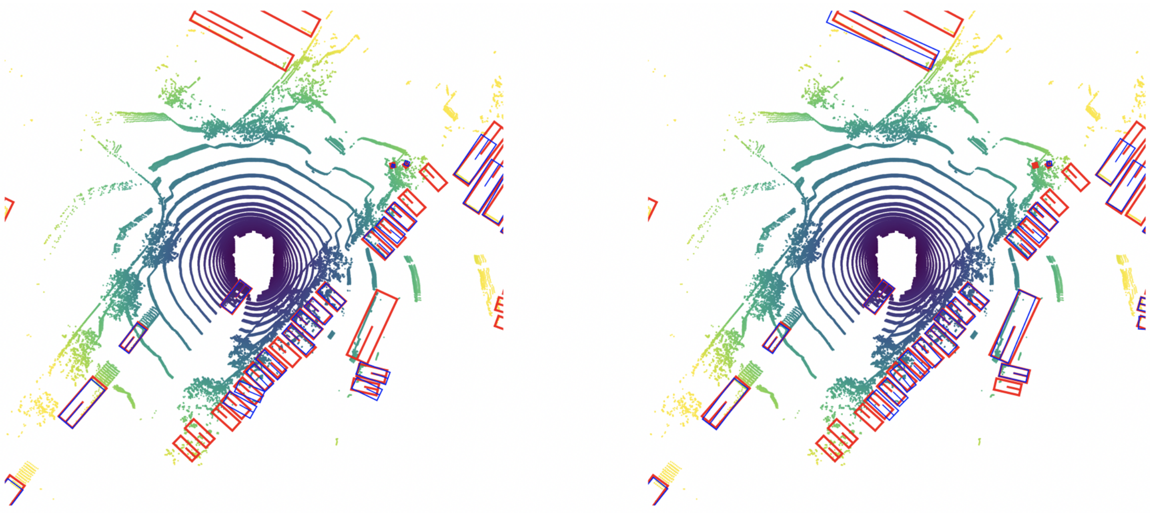


Fig. 3.11: Qualitative Comparison of detection results from CenterPoint (left) and Center Feature Fusion (right). Red boxes indicate ground-truth objects, while blue indicate model predictions. Our approach detects several cars and trucks that CenterPoint misses, as well as generating more accurate boxes for a number of objects.

approaches on the nuScenes test set. In addition to mAP and NDS, we also compare an approximate number of fused features per-frame. PointAugmenting [111] (and other related point-decoration methods) augment each point in the point cloud, which is, on average, around 200000 points in nuScenes after concatenation with non-key frames. TransFusion [6] calculates attention between LiDAR BEV features and CenterNet [138] camera features in perspective view, and so the fused features are aggregated across all more than 100000 pixels in the CenterNet output feature maps. BEVFusion [67] projects and fuses all camera features encoded by a Swin-T backbone, amounting to almost 150000 total pixels. In comparison, through our selective fusion approach, we toss out the majority of camera features and fetch only around 1000 features to project and fuse, more than 100× fewer than the aforementioned fusion methods. Despite this, we achieve comparable improvements in performance over the CenterPoint baseline. We outperform PointAugmenting on a handful of classes (e.g. bus, barrier) even without the test-time-augmentation PointAugmenting employs. On the more fine-grained classes of motorcyclists and bicyclist, which rely heavily on semantic information for detection, we achieve performance in-line with all of the other methods. We also perform a more direct comparison of inference speed, shown in Tab 3.3. As measured on an NVIDIA A6000 GPU, CFF achieves a latency speed of 5 FPS. In comparison, BEVFusion, a concurrent work also leveraging fusion in BEV space, is more than 5× slower at only 0.8 FPS. This shows the quantitative advantage gained from our feature-efficient approach to fusion.

	# of fused features	mAP	NDS	Car	Truck	C.V.	Bus	Trailer	Barrier	Motor.	Bike	Ped.	T.C.
CenterPoint	-	60.3	67.3	85.2	53.5	20.0	63.6	56.0	71.1	59.5	30.7	84.6	78.4
PointAugmenting	~200000	66.8	71.0	87.5	57.3	28.0	65.2	60.7	72.6	74.3	50.9	87.9	83.6
TransFusion	~100000	68.9	71.7	87.1	60.0	33.1	68.3	60.8	78.1	73.6	52.9	88.4	86.7
BEVFusion	~100000	**70.2**	**72.9**	88.6	60.1	39.3	69.8	63.8	80.0	74.1	51.0	89.2	86.5
Ours	**~1000**	65.2	69.9	84.7	55.4	26.0	66.2	59.2	74.3	72.6	50.7	85.7	77.5

Table 3.2: Comparison with state-of-the-art fusion methods and CenterPoint on nuScenes test set. Our method achieves a similar increase in performance to other fusion approaches, even as we use only approximately 1/100 the number of features for fusion.

Method	Inference Speed (FPS)
BEVFusion [60]	0.8
PointAugmenting [111]	2.9
Ours	**5.0**

Table 3.3: Inference speed comparison with concurrent SOTA fusion approaches. CFF increases inference speed by a factor of 2× to 5×. Inference speeds are measured on an NVIDIA A6000 GPU.

3.3.3 Ablation Studies

3.3.3.1 Augmentation Strategy

We validate the effectiveness of our ProjectionAlign data augmentation by comparing its performance against no augmentation, augmentation only the standard LiDAR augmentations are applied (i.e. camera and LiDAR features are misaligned), and our augmentation strategy. The results are summarized in Table 3.4. Without any sort of data augmentation, our fusion model quickly runs into overfitting and improvement over the LiDAR-only baseline is minimal at +0.8 mAP. Applying the naive augmentation strategy improves our model by significant amount of +2.8 mAP. By introducing our aligned augmentation strategy, the model performance is further improved by +1.2 mAP. This confirms the effectiveness of our augmentation scheme and the general importance of aligning fused features, as was shown in [58].

Augmentation Strategy	mAP	NDS
No Augmentation	60.4	66.6
+ LiDAR Only	63.2	68.2
+ ProjectionAlign	**64.5**	**69.2**

Table 3.4: Ablation study of ProjectionAlign. Results shown for nuScenes validation set.

3.3.3.2 Heatmap Thresholding

To illustrate the computational efficiency gained from our selective projection strategy, we perform experiments varying the heatmap threshold used to determine which camera features should be projected to BEV. These are summarized in Table 3.5. A threshold of 0, meaning every pixel across all 6 cameras is projected, requires on average more than 3.5 seconds on a single NVIDIA A6000 GPU. Given that most of the pixels are identified by the object detector as background, even a small threshold of 0.01 reduces the number of projected pixels and the required latency by more than 6×. We find that improvement in mAP levels off at a threshold of 0.1, which results in fusing almost 1/100 of the total camera features available, illustrating the redundancy of the vast majority of available features. Thus, we use the threshold of 0.1 in all of the experiments presented in this paper.

Threshold	# of projected pixels	Latency	mAP
0.5	110	57 ms	60.8
0.1	1442	74 ms	64.5
0.05	2855	98 ms	64.8
0.01	20077	558 ms	63.5
0.0	134400	3690 ms	-

Table 3.5: Comparison of average projection latency, number of pixels projected from perspective view to BEV and resulting mAP for various detection heatmap thresholds on nuScenes dataset. Projecting all camera pixels presents a significant processing bottleneck. We omit the mAP for a threshold of 0.0 as the latency makes training this model infeasible.

3.4 Discussion

In this chapter, we introduce a novel approach for fusing LiDAR and camera images for 3D object detection which we dub Center Feature Fusion (CFF). In contrast to previous fusion approaches, CFF fuses encoded sensor representations in bird's-eye-view space while simultaneously leveraging a center-based selective projection approach that allows for a significant latency improvement. We validate our approach on the nuScenes dataset, achieving a sizable +5 mAP improvement over the LiDAR-only baseline while performing competitively with competing fusion approaches, even while fusing a far smaller number of features and achieving faster inference speeds.

Although CFF achieves strong performance on the nuScenes dataset, several potential limitations/opportunities for improvements exist. One significant assumption CFF operates under is that the camera calibration parameters are available and accurate. Sensor calibration is a challenging problem itself, requiring a significant upfront effort for our fusion approach to have viability. Furthermore, in real operating conditions, the natural mechanical vibrations of a moving car can disrupt calibrated sensors, generating errors in the calibration parameters. As is, CFF has no mechanism to compensate for unpredictable errors in calibration. A related issue is sensor synchronization; misalignment in the times when camera and LiDAR sensors capture the scene can similarly disrupt the reliability of CFF. Another potential area of improvement for CFF is in the depth interpolation stage. While LiDAR-guided depth completion is substantially more accurate than monocular depth estimation, inaccuracies in per-pixel depth estimates still exist, and improved methods for interpolation are a direction of future investigation. An even more significant limitation to this approach is that depth estimation fails for objects either outside of the LiDAR range, or are too small to contain a significant number of points in the point cloud. For both of these cases, no depth estimation can be performed and fusion cannot be performed. Extending CFF to be capable of fusing information even outside of the LiDAR's range is a compelling problem, given the importance of long-range object detection in driving scenarios. Lastly, our current method for training CFF requires pretraining the camera-only CenterNet model separately, requiring significant extra compute be dedicated to this two-stage training process. Developing an approach to train CFF in an end-to-end manner is another possible direction for improving this work, especially considering the existence of fusion models which are trained end-to-end.

Even given these drawbacks, CFF introduces a meaningfully new approach to sensor fusion which performs competitively with previous literature approaches while offering a new analysis on how to improve the latency of the fusion mechanism. As is, CFF has the potential for implementation in real-world AV perception systems as an alternative to slower, more computationally intensive fusion-based object detectors.

Chapter 4

Doubly-Robust Self-Training

In this chapter, we will address 3D vision from the data efficiency angle. Based on theoretical analysis, we introduce a novel loss function, which we dub the doubly-robust loss, to be used for semi-supervised learning. On both image classification and object detection tasks, our loss function outperforms the baseline semi-supervised learning approach. This chapter is based on work previously published in [140].

4.1 Semi-Supervised Learning Overview

To circumvent the high data labeling costs associated with supervised learning, several different strategies have been proposed. Semi-supervised learning (SSL) is a popular machine learning paradigm that considers the problem of learning based on a small labeled dataset together with a large unlabeled dataset. This general framework plays an important role in many problems in machine learning, including model fine-tuning, model distillation, self-training, transfer learning and continual learning [142, 77, 117, 24, 20]. Similarly, semi-supervised learning has been widely adopted for computer vision, and particularly for the task of AV perception [54, 8, 7, 99, 119, 41, 85].

One popular approach to SSL, known as self-training or pseudo-labeling, works by first pre-training a *teacher* model in a supervised manner on the available labeled data. Subsequently, the teacher model is then used to generate predictions, dubbed *pseudo-labels*, on the unlabeled data. A new model, called a *student* model, is then trained on a combination of the labeled and pseudo-labeled data. A visualization of the self-training approach to SSL is shown in Fig. 4.1.

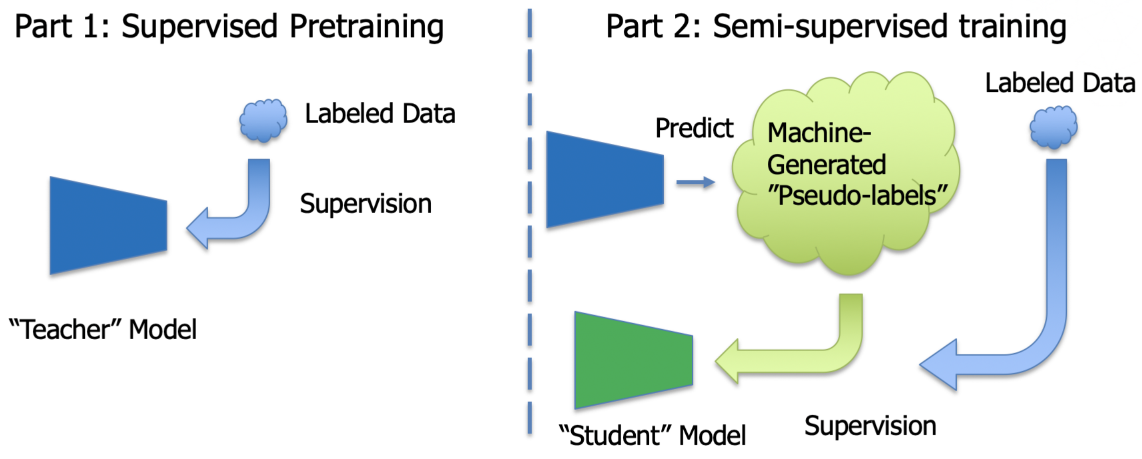

Part 1: Supervised Pretraining
Labeled Data
Supervision
"Teacher" Model
Part 2: Semi-supervised training
Labeled Data
Predict
Machine-Generated "Pseudo-labels"
"Student" Model
Supervision

Fig. 4.1: Overview of self-training SSL scheme. First, a teacher model is trained on the labeled data using supervised learning. The teacher model is then used in inference mode to generate pseudo-labels on the unlabeled data, and a student model is trained on a mixture of the labeled/pseudo-labeled data.

In contrast to fully supervised learning, pseudo-labeling relies on training the student model using inherently noisy labels due to inaccuracies in the teacher model predictions. Thus, to develop a strong pseudo-labeling scheme, two key challenges need to be addressed: how to extract the highest-quality pseduo-labels from the teacher model, and the best manner in which to use the pseudo-labeled data during student model training. Several strategies have been proposed to address these challenges that we will detail in the following sections.

4.1.1 Pseudo-labeling Approaches

Pseudo-labeling is a popular semi-supervised learning paradigm in which machine-generated pseudo-labels are used for training with unlabeled data [54, 8, 7, 99, 135]. To generate these pseudo-labels, a teacher model is pre-trained on a set of labeled data, and its predictions on the unlabeled data are extracted as pseudo-labels. Previous work seeks to address the noisy quality of pseudo-labels in various ways. Temporal Ensembling [47] uses a temporal averaging of pseudo-labels to improve their quality. Mean Teacher [103] updates the teacher model weights using an exponential moving average (EMA) with the student model weights such that the teacher model maintains an advantage over the student during training. MixMatch [8] ensembles pseudo-labels across several augmented views of the input data. ReMixMatch [7] extends this by weakly augmenting the teacher inputs and strongly augmenting the student inputs. FixMatch [99] uses confidence thresholding to select only high-quality pseudo-labels for student training.

4.1.2 Semi-supervised Learning for Vision

Self-training has been applied in both 2D computer vision problems [64, 39, 102, 100, 136] and 3D problems [79, 112, 56, 63] object detection. STAC [100] enforces consistency between strongly augmented versions of confidence-filtered pseudo-labels. Unbiased teacher [64] updates the teacher during training with an exponential moving average (EMA) of the student network weights. Dense Pseudo-Label [136] replaces box pseudo-labels with the raw output features of the detector to allow the student to learn richer context. In the 3D domain, 3DIoUMatch [112] thresholds pseudo-labels using a model-predicted Intersection-over-Union (IoU). DetMatch [79] performs detection in both the 2D and 3D domains and filters pseudo-labels based on 2D-3D correspondence. HSSDA [63] extends strong augmentation during training with a patch-based point cloud shuffling augmentation. Offboard3D [85] utilizes multiple frames of temporal context to improve pseudo-label quality.

4.2 Doubly-Robust Self-Training

In this section, we introduce our proposed doubly-robust loss function, and demonstrate through theoretical analysis its superiority over the naive semi-supervised loss function.

4.2.1 Overview

Our proposed doubly-robust loss is based on a simple modification of the standard loss for self-training. Assume that we are given a set of unlabeled samples, $\mathcal{D}_1 = \{X_1, X_2, \cdots, X_m\}$, drawn from a fixed distribution $\mathbb{P}_X$, a set of labeled samples $\mathcal{D}_2 = \{(X_{m+1}, Y_{m+1}), (X_{m+2}, Y_{m+2}), \cdots, (X_{m+n}, Y_{m+n})\}$, drawn from some joint distribution $\mathbb{P}_X \times \mathbb{P}_{Y|X}$, and a teacher model $\hat{f}$. Our target is to find some $\theta^\star \in \Theta$ that satisfies

$$\theta^\star \in \arg\min_{\theta\in\Theta} \mathbb{E}_{(X,Y)\sim\mathbb{P}_X\times\mathbb{P}_{Y|X}}[\ell_\theta(X, Y)].$$

Let $\ell_\theta(x, y)$ be a pre-specified loss function that characterizes the prediction error of the estimator with parameter θ on the given sample (X, Y). Consider a naive estimator that ignores the predictor $\hat{f}$ and only trains on the labeled samples:

$$\mathcal{L}^{\mathsf{TL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, Y_i).$$

Although naive, this is a safe choice since it is an empirical risk minimizer. As $n \to \infty$, the loss converges to the population loss. However, it ignores all the information provided in $\hat{f}$ and the unlabeled dataset, which makes it inefficient when the predictor $\hat{f}$ is informative. On the other hand, traditional self-training aims at minimizing the combined loss for both labeled and unlabeled samples, where the pseudo-labels for unlabeled samples are generated using $\hat{f}$[1]:

$$\mathcal{L}^{\mathsf{SL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m+n}\left(\sum_{i=1}^{m} \ell_\theta(X_i, \hat{f}(X_i)) + \sum_{i=m+1}^{m+n} \ell_\theta(X_i, Y_i)\right).$$

Note that this can also be viewed as first using $\hat{f}$ to predict all the data, and then replacing the originally labeled points with the known labels:

$$\mathcal{L}^{\mathsf{SL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m+n}\sum_{i=1}^{m+n} \ell_\theta(X_i, \hat{f}(X_i)) - \frac{1}{m+n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, \hat{f}(X_i)) + \frac{1}{m+n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, Y_i).$$

As is shown by the last equality, the self-training loss can be viewed as first using $\hat{f}$ to predict all the samples (including the labeled samples) and computing the average loss, then replacing that part of the loss corresponding to the labeled samples with the loss on the original labels. Although the loss uses the information arising from the unlabeled samples and $\hat{f}$, the performance can be poor when the predictor is not accurate. Our proposed doubly-robust loss instead replaces the coefficient $1/(m+n)$ with $1/n$ in the last two terms:

$$\mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m+n}\sum_{i=1}^{m+n} \ell_\theta(X_i, \hat{f}(X_i)) - \frac{1}{n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, \hat{f}(X_i)) + \frac{1}{n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, Y_i). \tag{4.1}$$

This seemingly minor change has a major beneficial effect—the estimator becomes consistent and doubly robust.

[1]There are several variants of the traditional self-training loss. For example, [119] introduce an extra weight $(m + n)/n$ on the labeled samples, and add noise to the student model; [99] use confidence thresholding to filter unreliable pseudo-labels. However, both of these alternatives still suffer from the inconsistency issue. In this chapter we focus on the simplest form $\mathcal{L}^{\mathsf{SL}}$.

Theorem 1 (Informal) *Let $\theta^\star$ be defined as the minimizer $\theta^\star = \arg\min_\theta \mathbb{E}_{(X,Y)\sim\mathbb{P}_X\times\mathbb{P}_{Y|X}}[\ell_\theta(X,Y)]$. Under certain regularity conditions, we have*

$$\|\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)\|_2 \lesssim \begin{cases} \sqrt{\frac{d}{m+n}}, & \textit{when } Y \equiv \hat{f}(X), \\ \sqrt{\frac{d}{n}}, & \textit{otherwise.} \end{cases}$$

On the other hand, there exists instances such that $\|\nabla_\theta \mathcal{L}^{\mathsf{SL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)\|_2 \geq C$ always holds true no matter how large m, n are.

The result shows that the true parameter $\theta^\star$ is also a local minimum of the doubly robust loss, but not a local minimum of the original self-training loss. We provide an intuitive interpretation here (including a visualization in Fig. 4.2):

- In the case when the given predictor is perfectly accurate, i.e., $\hat{f}(X) \equiv Y$ always holds (which also means that $Y|X = x$ is a deterministic function of x), the last two terms cancel, and the loss minimizes the average loss, $\frac{1}{m+n}\sum_{i=1}^{m+n}\ell_\theta(X_i, \hat{f}(X_i))$, on all of the provided data. The effective sample size is $m+n$, compared with effective sample size n for training only on a labeled dataset using $\mathcal{L}^{\mathsf{TL}}$. In this case, the loss $\mathcal{L}^{\mathsf{DR}}$ is much better than $\mathcal{L}^{\mathsf{TL}}$, and comparable to $\mathcal{L}^{\mathsf{SL}}$. We may as well relax the assumption of $\hat{f}(X) = Y$ to $\mathbb{E}[\ell_\theta(X, \hat{f}(X))] = \mathbb{E}[\ell_\theta(X, Y)]$. As n grows larger, the loss is approximately minimizing the average loss $\frac{1}{m+n}\sum_{i=1}^{m+n}\ell_\theta(X_i, \hat{f}(X_i))$.
- On the other hand, no matter how bad the given predictor is, the difference between the first two terms vanishes as either of m, n goes to infinity since the labeled samples $X_{m+1}, \cdots, X_{m+n}$ arise from the same distribution as $X_1, \cdots, X_m$. Thus asymptotically the loss minimizes $\frac{1}{n}\sum_{i=m+1}^{m+n}\ell_\theta(X_i, Y_i)$, which discards the bad predictor $\hat{f}$ and focuses only on the labeled dataset. Thus, in this case the loss $\mathcal{L}^{\mathsf{DR}}$ is much better than $\mathcal{L}^{\mathsf{SL}}$, and comparable to $\mathcal{L}^{\mathsf{TL}}$.

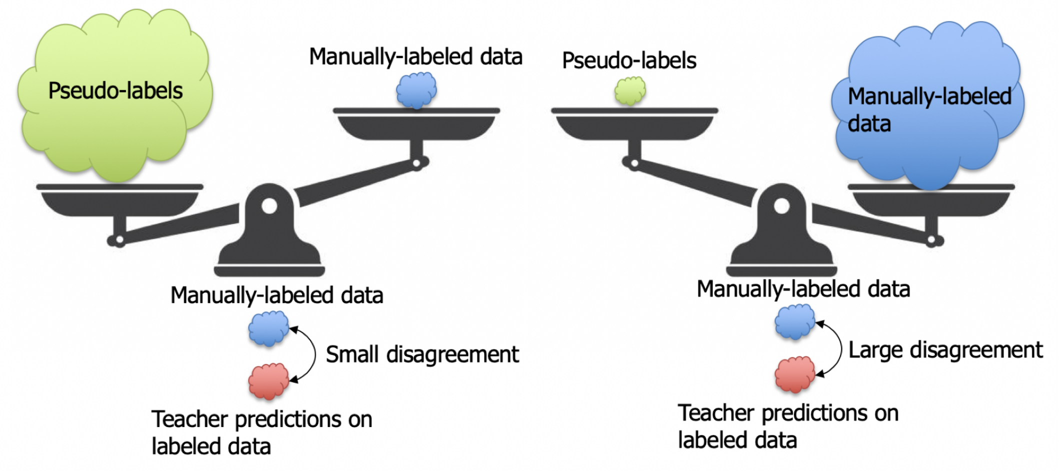


Fig. 4.2: Visualization of intuitive function of the doubly-robust loss. When the teacher model is perfectly accurate, the loss heavily weights the contribution from pseudo-labeled data. Meanwhile if the teacher model is a poor predictor, the loss focuses on the contribution from the labeled data.

This loss is appropriate only when the covariate distributions between labeled and unlabeled samples match. In the case where there is a distribution mismatch, we propose an alternative loss; see Section 4.2.4.

4.2.2 Motivating example: Mean estimation

As a concrete example, in the case of one-dimensional mean estimation we take $\ell_\theta(X, Y) = (\theta - Y)^2$. Our target is to find some $\theta^\star$ that satisfies

$$\theta^\star = \arg\min_\theta \mathbb{E}_{(X,Y)\sim\mathbb{P}_X\times\mathbb{P}_{Y|X}}[(\theta - Y)^2].$$

One can see that $\theta^\star = \mathbb{E}[Y]$. In this case, the loss for training only on labeled data becomes

$$\mathcal{L}^{\mathsf{TL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{n}\sum_{i=m+1}^{m+n}(\theta - Y_i)^2.$$

Moreover, the optimal parameter is $\hat{\theta}_{\mathsf{TL}} = \frac{1}{n}\sum_{i=m+1}^{m+n} Y_i$, which is a simple empirical average over all observed Y's.

For a given pre-existing predictor $\hat{f}$, the loss for self-training becomes

$$\mathcal{L}^{\mathsf{SL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m+n}\left(\sum_{i=1}^{m}(\theta - \hat{f}(X_i))^2 + \sum_{i=m+1}^{m+n}(\theta - Y_i)^2\right).$$

It is straightforward to see that the minimizer of the loss is the unweighted average between the unlabeled predictors $\hat{f}(X_i)$'s and the labeled Y_i's:

$$\theta^\star_{\mathsf{SL}} = \frac{1}{m+n}\left(\sum_{i=1}^{m}\hat{f}(X_i) + \sum_{i=m+1}^{m+n} Y_i\right).$$

In the case of $m \gg n$, the mean estimator is almost the same as the average of all the predicted values on the unlabeled dataset, which can be far from $\theta^\star$ when the predictor $\hat{f}$ is inaccurate.

On the other hand, for the proposed doubly robust estimator, we have

$$\begin{aligned}\mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) &= \frac{1}{m+n}\sum_{i=1}^{m+n}(\theta - \hat{f}(X_i))^2 - \frac{1}{n}\sum_{i=m+1}^{m+n}(\theta - \hat{f}(X_i))^2 + \frac{1}{n}\sum_{i=m+1}^{m+n}(\theta - Y_i)^2 \\ &= \frac{1}{m+n}\sum_{i=1}^{m+n}(\theta - \hat{f}(X_i))^2 + \frac{1}{n}\sum_{i=m+1}^{m+n} 2(\hat{f}(X_i) - Y_i)\theta + Y_i^2 - \hat{f}(X_i)^2.\end{aligned}$$

Note that the loss is still convex, and we have

$$\theta^\star_{\mathsf{DR}} = \frac{1}{m+n}\sum_{i=1}^{m+n}\hat{f}(X_i) - \frac{1}{n}\sum_{i=m+1}^{m+n}(\hat{f}(X_i) - Y_i).$$

This recovers the estimator in prediction-powered inference[2]. Assume that $\hat{f}$ is independent of the labeled data. We can calculate the mean-squared error of the three estimators as follows.

Proposition 1 *Let* $\mathsf{Var}[\hat{f}(X) - Y] = \mathbb{E}[(\hat{f}(X) - Y)^2 - \mathbb{E}[(\hat{f}(X) - Y)]^2]$. *We have*

$$\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{TL}})^2] = \frac{1}{n}\mathsf{Var}[Y],$$

$$\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{SL}})^2] \leq \frac{2m^2}{(m+n)^2}\mathbb{E}[(\hat{f}(X) - Y)]^2 + \frac{2m}{(m+n)^2}\mathsf{Var}[\hat{f}(X) - Y] + \frac{2n}{(m+n)^2}\mathsf{Var}[Y],$$

$$\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{DR}})^2] \leq 2\min\left(\frac{1}{n}\mathsf{Var}[Y] + \frac{m+2n}{(m+n)n}\mathsf{Var}[\hat{f}(X)], \frac{m+2n}{(m+n)n}\mathsf{Var}[\hat{f}(X) - Y] + \frac{1}{m+n}\mathsf{Var}[Y]\right).$$

As proof, consider the following: for the labeled-only estimator $\hat{\theta}_{\mathsf{TL}}$, we have

$$\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{TL}})^2] = \mathbb{E}\left[\left(\mathbb{E}[Y] - \frac{1}{n}\sum_{i=m+1}^{m+n} Y\right)^2\right] = \frac{1}{n}\mathsf{Var}[Y].$$

For the self-training loss, we have

$$\begin{aligned}
\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{SL}})^2] &= \mathbb{E}\left[\left(\mathbb{E}[Y] - \frac{1}{m+n}\left(\sum_{i=1}^{m}\hat{f}(X_i) + \sum_{i=m+1}^{m+n} Y_i\right)\right)^2\right] \\
&\leq 2\left(\mathbb{E}\left[\left(\frac{m}{m+n}\left(\mathbb{E}[Y] - \frac{1}{m}\sum_{i=1}^{m}\hat{f}(X_i)\right)\right)^2\right] + \mathbb{E}\left[\left(\frac{n}{m+n}\left(\mathbb{E}[Y] - \frac{1}{n}\sum_{i=m+1}^{m+n} Y_i\right)\right)^2\right]\right) \\
&\leq \frac{2m^2}{(m+n)^2}\mathbb{E}[(\hat{f}(X) - Y)]^2 + \frac{2m}{(m+n)^2}\mathsf{Var}[\hat{f}(X) - Y] + \frac{2n}{(m+n)^2}\mathsf{Var}[Y].
\end{aligned}$$

For the doubly robust loss, on one hand, we have

$$\begin{aligned}
\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{DR}})^2] &= \mathbb{E}\left[\left(\mathbb{E}[Y] - \frac{1}{m+n}\sum_{i=1}^{m+n}\hat{f}(X_i) + \frac{1}{n}\sum_{i=m+1}^{m+n}(\hat{f}(X_i) - Y_i)\right)^2\right] \\
&\leq 2\mathbb{E}\left[\left(\mathbb{E}[Y] - \frac{1}{n}\sum_{i=m+1}^{m+n} Y_i\right)^2\right] + 2\mathbb{E}\left[\left(\mathbb{E}[\hat{f}(X)] - \frac{1}{n}\sum_{i=m+1}^{m+n}\hat{f}(X_i)\right)^2\right] \\
&\quad + 2\mathbb{E}\left[\left(\mathbb{E}[\hat{f}(X)] - \frac{1}{m+n}\sum_{i=1}^{m+n}\hat{f}(X_i)\right)^2\right] \\
&= \frac{2}{n}\mathsf{Var}[Y] + \left(\frac{2}{m+n} + \frac{2}{n}\right)\mathsf{Var}[\hat{f}(X)].
\end{aligned}$$

On the other hand, we have

$$\mathbb{E}[(\theta^\star - \hat{\theta}_{\mathsf{DR}})^2] = \mathbb{E}\left[\left(\mathbb{E}[Y] - \frac{1}{m+n}\sum_{i=1}^{m+n}\hat{f}(X_i) + \frac{1}{n}\sum_{i=m+1}^{m+n}(\hat{f}(X_i) - Y_i)\right)^2\right]$$

$$\leq 2\mathbb{E}\left[\left(\mathbb{E}[Y] - \frac{1}{m+n}\sum_{i=1}^{m+n}Y_i\right)^2\right] + 2\mathbb{E}\left[\left(\mathbb{E}[\hat{f}(X) - Y] - \frac{1}{n}\sum_{i=m+1}^{m+n}(\hat{f}(X_i) - Y_i)\right)^2\right]$$

$$+ 2\mathbb{E}\left[\left(\mathbb{E}[\hat{f}(X) - Y] - \frac{1}{m+n}\sum_{i=1}^{m+n}(\hat{f}(X_i) - Y_i)\right)^2\right]$$

$$= \left(\frac{2}{m+n} + \frac{2}{n}\right)\mathsf{Var}[\hat{f}(X) - Y] + \frac{2}{m+n}\mathsf{Var}[Y].$$

The proof is done by taking the minimum of the two upper bounds. The proposition illustrates the double-robustness of $\hat{\theta}_{\mathsf{DR}}$—no matter how poor the estimator $\hat{f}(X)$ is, the rate is always upper bounded by $\frac{4}{n}(\mathsf{Var}[Y] + \mathsf{Var}[\hat{f}(X)])$. On the other hand, when $\hat{f}(X)$ is an accurate estimator of Y (i.e., $\mathsf{Var}[\hat{f}(X) - Y]$ is small), the rate can be improved to $\frac{2}{m+n}\mathsf{Var}[Y]$. In contrast, the self-training loss always has a non-vanishing term, $\frac{2m^2}{(m+n)^2}\mathbb{E}[(\hat{f}(X) - Y)]^2$, when $m \gg n$, unless the predictor $\hat{f}$ is accurate.

On the other hand, when $\hat{f}(x) = \hat{\beta}_{(-1)}^\top x + \hat{\beta}_1$ is a linear predictor trained on the labeled data with $\hat{\beta} = \arg\min_{\beta=[\beta_1,\beta_{(-1)}]}\frac{1}{n}\sum_{i=m+1}^{m+n}(\beta_{(-1)}^\top X_i + \beta_1 - Y_i)^2$, our estimator reduces to the semi-supervised mean estimator in[132]. Let $\tilde{X} = [1, X]$. In this case, we also know that the self-training reduces to training only on labeled data, since $\hat{\theta}_{\mathsf{TL}}$ is also the minimizer of the self-training loss. We have the following result that reveals the superiority of the doubly robust estimator compared to the other two options.

Proposition 2 ([132]) *We establish the asymptotic behavior of various estimators when $\hat{f}$ is a linear predictor trained on the labeled data:*

- *Training only on labeled data $\hat{\theta}_{\mathsf{TL}}$ is equivalent to self-training $\hat{\theta}_{\mathsf{SL}}$, which gives unbiased estimator but with larger variance:*

$$\sqrt{n}(\hat{\theta}_{\mathsf{TL}} - \theta^\star) \to \mathcal{N}(0, \mathbb{E}[(Y - \beta^\top\tilde{X})^2] + \beta_{(-1)}^\top\Sigma\beta_{(-1)}).$$

- *Doubly Robust $\hat{\theta}_{\mathsf{DR}}$ is unbiased with smaller variance:*

$$\sqrt{n}(\hat{\theta}_{\mathsf{DR}} - \theta^\star) \to \mathcal{N}(0, \mathbb{E}[(Y - \beta^\top\tilde{X})^2] + \frac{n}{m+n}\beta_{(-1)}^\top\Sigma\beta_{(-1)}).$$

Here $\beta = \arg\min_\beta \mathbb{E}[(Y - \beta^\top\tilde{X})^2]$ and $\Sigma = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top]$.

4.2.3 Guarantee for general loss

In the general case, we show that the doubly robust loss function continues to exhibit desirable properties. In particular, as n, m goes to infinity, the global minimum of the original loss is also a critical point of the new doubly robust loss, no matter how inaccurate the predictor $\hat{f}$.

Let $\theta^\star$ be the minimizer of $\mathbb{E}_{\mathbb{P}_{X,Y}}[\ell_\theta(X,Y)]$. Let $\hat{f}$ be a pre-existing model that does not depend on the datasets $\mathcal{D}_1, \mathcal{D}_2$. We also make the following regularity assumptions.

Assumption 1 *The loss $\ell_\theta(x,y)$ is differentiable at $\theta^\star$ for any x, y.*

Assumption 2 *The random variables $\nabla_\theta \ell_\theta(X, \hat{f}(X))$ and $\nabla_\theta \ell_\theta(X,Y)$ have bounded first and second moments.*

Given this assumption, we denote $\Sigma_\theta^{Y-\hat{f}} = \mathsf{Cov}[\nabla_\theta \ell_\theta(X,\hat{f}(X)) - \nabla_\theta \ell_\theta(X,Y)]$ and let $\Sigma_\theta^{\hat{f}} = \mathsf{Cov}[\nabla_\theta \ell_\theta(X,\hat{f}(X))]$, $\Sigma_\theta^{Y} = \mathsf{Cov}[\nabla_\theta \ell_\theta(X,Y)]$.

Theorem 2 *Under Assumptions 1 and 2, we have that with probability at least $1-\delta$,*

$$\|\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)\|_2 \le C \min\Bigg(\|\Sigma_{\theta^\star}^{\hat{f}}\|_2 \sqrt{\frac{d}{(m+n)\delta}} + \|\Sigma_{\theta^\star}^{Y-\hat{f}}\|_2 \sqrt{\frac{d}{n\delta}}, \\ \|\Sigma_{\theta^\star}^{\hat{f}}\|_2 \left(\sqrt{\frac{d}{(m+n)\delta}} + \sqrt{\frac{d}{n\delta}} \right) + \|\Sigma_{\theta^\star}^{Y}\|_2 \sqrt{\frac{d}{n\delta}} \Bigg),$$

where C is a universal constant, and $\mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}$ is defined in Equation (4.1).

As proof, consider the following: we know that

$$\begin{aligned}
&\|\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star) - \mathbb{E}[\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)]\|_2 \\
&= \Big\| \frac{1}{m+n} \sum_{i=1}^{m+n} (\nabla_\theta \ell_{\theta^\star}(X_i, \hat{f}(X_i)) - \mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X, \hat{f}(X))]) + \\
&\qquad \frac{1}{n} \sum_{i=m+1}^{m+n} \Big(\nabla_\theta \ell_{\theta^\star}(X_i, Y_i) - \nabla_\theta \ell_{\theta^\star}(X_i, \hat{f}(X_i)) \\
&\qquad\qquad - \mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,Y) - \nabla_\theta \ell_{\theta^\star}(X, \hat{f}(X))] \Big) \Big\|_2 \\
&\le \Big\| \frac{1}{m+n} \sum_{i=1}^{m+n} (\nabla_\theta \ell_{\theta^\star}(X_i, \hat{f}(X_i)) - \mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X, \hat{f}(X))]) \Big\|_2 + \\
&\qquad \Big\| \frac{1}{n} \sum_{i=m+1}^{m+n} \Big(\nabla_\theta \ell_{\theta^\star}(X_i, Y_i) - \nabla_\theta \ell_{\theta^\star}(X_i, \hat{f}(X_i)) \\
&\qquad\qquad - \mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,Y) - \nabla_\theta \ell_{\theta^\star}(X, \hat{f}(X))] \Big) \Big\|_2 .
\end{aligned}$$

From the multi-dimensional Chebyshev inequality[9, 73], we have that with probability at least $1-\delta/2$, for some universal constant C,

$$\Big\| \frac{1}{m+n} \sum_{i=1}^{m+n} (\nabla_\theta \ell_{\theta^\star}(X_i, \hat{f}(X_i)) - \mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X, \hat{f}(X))]) \Big\|_2 \le C \|\Sigma_{\theta^\star}^{\hat{f}}\|_2 \sqrt{\frac{d}{(m+n)\delta}}.$$

Similarly, we also have that with probability at least $1-\delta/2$,

$$\Big\|\frac{1}{n}\sum_{i=m+1}^{m+n}\Big(\nabla_\theta \ell_{\theta^\star}(X_i,Y_i)-\nabla_\theta \ell_{\theta^\star}(X_i,\hat{f}(X_i))-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,Y)-\nabla_\theta \ell_{\theta^\star}(X,\hat{f}(X))]\Big)\Big\|_2 \leq C\|\Sigma_{\theta^\star}^{Y-\hat{f}}\|_2\sqrt{\frac{d}{n\delta}}.$$

Furthermore, note that

$$\mathbb{E}[\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)] = \mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,Y)] = \nabla_\theta \mathbb{E}[\ell_{\theta^\star}(X,Y)] = 0.$$

Here we use Assumption 1 and Assumption 2 to ensure that the expectation and differentiation are interchangeable. Thus we have that with probability at least $1-\delta$,

$$\|\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)\|_2 \leq C\left(\|\Sigma_{\theta^\star}^{\hat{f}}\|_2\sqrt{\frac{d}{(m+n)\delta}}+\|\Sigma_{\theta^\star}^{Y-\hat{f}}\|_2\sqrt{\frac{d}{n\delta}}\right).$$

On the other hand, we can also write the difference as

$$\begin{aligned}
&\|\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)-\mathbb{E}[\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)]\|_2\\
&\quad= \Big\|\frac{1}{m+n}\sum_{i=1}^{m+n}(\nabla_\theta \ell_{\theta^\star}(X_i,\hat{f}(X_i))-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,\hat{f}(X))])\\
&\qquad+\frac{1}{n}\sum_{i=m+1}^{m+n}\Big(\nabla_\theta \ell_{\theta^\star}(X_i,Y_i)-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,Y)]\Big)\\
&\qquad-\frac{1}{n}\sum_{i=m+1}^{m+n}\Big(\nabla_\theta \ell_{\theta^\star}(X_i,Y_i)-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,\hat{f}(X))]\Big)\Big\|_2\\
&\quad\leq \Big\|\frac{1}{m+n}\sum_{i=1}^{m+n}(\nabla_\theta \ell_{\theta^\star}(X_i,\hat{f}(X_i))-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,\hat{f}(X))])\Big\|_2\\
&\qquad+\Big\|\frac{1}{n}\sum_{i=m+1}^{m+n}\Big(\nabla_\theta \ell_{\theta^\star}(X_i,Y_i)-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,Y)]\Big)\Big\|_2\\
&\qquad+\Big\|\frac{1}{n}\sum_{i=m+1}^{m+n}\Big(\nabla_\theta \ell_{\theta^\star}(X_i,Y_i)-\mathbb{E}[\nabla_\theta \ell_{\theta^\star}(X,\hat{f}(X))]\Big)\Big\|_2\\
&\quad\leq C\left(\|\Sigma_{\theta^\star}^{\hat{f}}\|_2\left(\sqrt{\frac{d}{(m+n)\delta}}+\sqrt{\frac{d}{n\delta}}\right)+\|\Sigma_{\theta^\star}^{Y}\|_2\sqrt{\frac{d}{n\delta}}\right).
\end{aligned}$$

Here the last inequality uses the multi-dimensional Chebyshev inequality and it holds with probability at least $1-\delta$. This finishes the proof.

From the example of mean estimation we know that one can design instances such that $\|\nabla_\theta \mathcal{L}^{\mathsf{SL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star)\|_2 \geq C$ for some positive constant C.

When the loss $\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}$ is convex, the global minimum of $\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}$ converges to $\theta^\star$ as both m, n go to infinity. When the loss $\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}$ is strongly convex, it also implies that $\|\hat{\theta} - \theta^\star\|_2$ converges to zero as both m, n go to infinity, where $\hat{\theta}$ is the minimizer of $\nabla_\theta \mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}$.

When $\hat{f}$ is a perfect predictor with $\hat{f}(X) \equiv Y$ (and $Y|X = x$ is deterministic), one has $\mathcal{L}^{\mathsf{DR}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta^\star) = \frac{1}{m+n}\sum_{i=1}^{m+n} \ell_\theta(X_i, Y_i)$. The effective sample size is $m+n$ instead of n in $\mathcal{L}^{\mathsf{SL}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta)$.

When $\hat{f}$ is also trained from the labeled data, one may apply data splitting to achieve the same guarantee up to a constant factor. In Theorem 2, we analyzed the double robustness of the proposed loss function when the predictor $\hat{f}$ is pre-existing and not trained from the labeled dataset. In practice, one may only have access to the labeled and unlabeled datasets without a pre-existing teacher model. In this case, one may choose to split the labeled samples $\mathcal{D}_2$ into two parts. The last $n/2$ samples are used to train $\hat{f}$, and the first $n/2$ samples are used in the doubly robust loss:

$$\mathcal{L}^{\mathsf{DR2}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell_\theta(X_i, \hat{f}(X_i)) - \frac{2}{n}\sum_{i=m+1}^{m+n/2} \frac{1}{\pi(X_i)}\ell_\theta(X_i, \hat{f}(X_i)) + \frac{2}{n}\sum_{i=m+1}^{m+n/2} \frac{1}{\pi(X_i)}\ell_\theta(X_i, Y_i).$$

Since $\hat{f}$ is independent of all samples used in the above loss, the result in Theorem 2 continues to hold. Asymptotically, such a doubly robust estimator is never worse than the estimator trained only on the labeled data.

4.2.4 The case of distribution mismatch

We also consider the case in which the marginal distributions of the covariates for the labeled and unlabeled datasets are different. Assume in particular that we are given a set of unlabeled samples, $\mathcal{D}_1 = \{X_1, X_2, \cdots, X_m\}$, drawn from a fixed distribution $\mathbb{P}_X$, a set of labeled samples, $\mathcal{D}_2 = \{(X_{m+1}, Y_{m+1}), (X_{m+2}, Y_{m+2}), \cdots, (X_{m+n}, Y_{m+n})\}$, drawn from some joint distribution $\mathbb{Q}_X \times \mathbb{P}_{Y|X}$, and a pre-trained model $\hat{f}$. In the case when the labeled samples do not follow the same distribution as the unlabeled samples, we need to introduce an importance weight $\pi(x)$. This yields the following doubly robust estimator:

$$\mathcal{L}^{\mathsf{DR2}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell_\theta(X_i, \hat{f}(X_i)) - \frac{1}{n}\sum_{i=m+1}^{m+n} \frac{1}{\pi(X_i)}\ell_\theta(X_i, \hat{f}(X_i)) + \frac{1}{n}\sum_{i=m+1}^{m+n} \frac{1}{\pi(X_i)}\ell_\theta(X_i, Y_i).$$

Note that we not only introduce the importance weight π, but we also change the first term from the average of all the $m+n$ samples to the average of n samples.

Proposition 3 *We have $\mathbb{E}[\mathcal{L}^{\mathsf{DR2}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta)] = \mathbb{E}_{\mathbb{P}_{X,Y}}[\ell_\theta(X, Y)]$ as long as one of the following two assumptions hold:*

- *For any x, $\pi(x) = \frac{\mathbb{P}_X(x)}{\mathbb{Q}_X(x)}$.*
- *For any x, $\ell_\theta(x, \hat{f}(x)) = \mathbb{E}_{Y\sim\mathbb{P}_{Y|X=x}}[\ell_\theta(x, Y)]$.*

To prove this proposition, consider the following: we have

$$\begin{aligned}
\mathbb{E}[\mathcal{L}^{\mathsf{DR2}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta)] &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{X_i\sim\mathbb{P}_X}[\ell_\theta(X_i,\hat{f}(X_i))] - \frac{1}{n}\sum_{i=m+1}^{m+n}\mathbb{E}_{X_i\sim\mathbb{Q}_X}\left[\frac{1}{\pi(X_i)}\ell_\theta(X_i,\hat{f}(X_i))\right] \\
&\quad + \frac{1}{n}\sum_{i=m+1}^{m+n}\mathbb{E}_{X_i\sim\mathbb{Q}_X,Y_i\sim\mathbb{P}_{Y|X_i}}\left[\frac{1}{\pi(X_i)}\ell_\theta(X_i,Y_i)\right] \\
&= \mathbb{E}_{X\sim\mathbb{P}_X}[\ell_\theta(X,\hat{f}(X))] - \mathbb{E}_{X\sim\mathbb{Q}_X}\left[\frac{1}{\pi(X)}\ell_\theta(X,\hat{f}(X))\right] \\
&\quad + \mathbb{E}_{X\sim\mathbb{Q}_X,Y\sim\mathbb{P}_{Y|X}}\left[\frac{1}{\pi(X)}\ell_\theta(X,Y)\right].
\end{aligned}$$

In the first case when $\pi(x)\equiv\frac{\mathbb{P}_X(x)}{\mathbb{Q}_X(x)}$, we have

$$\begin{aligned}
\mathbb{E}[\mathcal{L}^{\mathsf{DR2}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta)] &= \mathbb{E}_{X\sim\mathbb{P}_X}[\ell_\theta(X,\hat{f}(X))] - \mathbb{E}_{X\sim\mathbb{Q}_X}\left[\frac{\mathbb{P}_X(X)}{\mathbb{Q}_X(X)}\ell_\theta(X,\hat{f}(X))\right] \\
&\quad + \mathbb{E}_{X\sim\mathbb{Q}_X,Y\sim\mathbb{P}_{Y|X}}\left[\frac{\mathbb{P}_X(X)}{\mathbb{Q}_X(X)}\ell_\theta(X,Y)\right] \\
&= \mathbb{E}_{X\sim\mathbb{P}_X}[\ell_\theta(X,\hat{f}(X))] - \mathbb{E}_{X\sim\mathbb{P}_X}\left[\ell_\theta(X,\hat{f}(X))\right] \\
&\quad + \mathbb{E}_{X\sim\mathbb{P}_X,Y\sim\mathbb{P}_{Y|X}}\left[\ell_\theta(X,Y)\right] \\
&= \mathbb{E}_{X,Y\sim\mathbb{P}_{X,Y}}\left[\ell_\theta(X,Y)\right].
\end{aligned}$$

In the second case when $\ell_\theta(x,\hat{f}(x)) = \mathbb{E}_{Y\sim\mathbb{P}_{Y|X=x}}[\ell_\theta(x,Y)]$, we have

$$\begin{aligned}
\mathbb{E}[\mathcal{L}^{\mathsf{DR2}}_{\mathcal{D}_1,\mathcal{D}_2}(\theta)] &= \mathbb{E}_{X\sim\mathbb{P}_X}[\ell_\theta(X,\hat{f}(X))] - \mathbb{E}_{X\sim\mathbb{Q}_X}\left[\frac{1}{\pi(X)}\ell_\theta(X,\hat{f}(X))\right] \\
&\quad + \mathbb{E}_{X\sim\mathbb{Q}_X}\mathbb{E}_{Y\sim\mathbb{P}_{Y|X}}\left[\frac{1}{\pi(X)}\ell_\theta(X,Y)\mid X\right] \\
&= \mathbb{E}_{X\sim\mathbb{P}_X}[\ell_\theta(X,\hat{f}(X))] - \mathbb{E}_{X\sim\mathbb{Q}_X}\left[\frac{1}{\pi(X)}\ell_\theta(X,\hat{f}(X))\right] \\
&\quad + \mathbb{E}_{X\sim\mathbb{Q}_X}\left[\frac{1}{\pi(X)}\ell_\theta(X,\hat{f}(X))\right] \\
&= \mathbb{E}_{X\sim\mathbb{P}_X}[\ell_\theta(X,\hat{f}(X))] \\
&= \mathbb{E}_{X,Y\sim\mathbb{P}_{X,Y}}\left[\ell_\theta(X,Y)\right].
\end{aligned}$$

This finishes the proof. The proposition implies that as long as either π or $\hat{f}$ is accurate, the expectation of the loss is the same as that of the target loss. When the distributions for the unlabeled and labeled samples match each other, this reduces to the case in the previous sections. In this case, taking $\pi(x)=1$ guarantees that the expectation of the doubly robust loss is always the same as that of the target loss.

4.3 Experimental Results

To employ the new doubly robust loss in practical applications, we need to specify an appropriate optimization procedure, in particular one that is based on (mini-batched) stochastic gradient descent so as to exploit modern scalable machine learning methods. In preliminary experiments we observed that directly minimizing the doubly robust loss in Equation (4.1) with stohastic gradient can lead to instability, and thus, we propose instead to minimize the curriculum-based loss in each epoch:

$$\mathcal{L}^{\mathsf{DR},t}_{\mathcal{D}_1,\mathcal{D}_2}(\theta) = \frac{1}{m+n}\sum_{i=1}^{m+n} \ell_\theta(X_i, \hat{f}(X_i)) - \alpha_t \cdot \left(\frac{1}{n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, \hat{f}(X_i)) - \frac{1}{n}\sum_{i=m+1}^{m+n} \ell_\theta(X_i, Y_i) \right).$$

As we show in the experiments below, this choice yields a stable algorithm. We set $\alpha_t = t/T$, where T is the total number of epochs. For the object detection experiments, we introduce the labeled samples only in the final epoch, setting $\alpha_t = 0$ for all epochs before setting $\alpha_t = 1$ in the final epoch. Intuitively, we start from the training with samples only from the pseudo-labels, and gradually introduce the labeled samples in the doubly robust loss for fine-tuning.

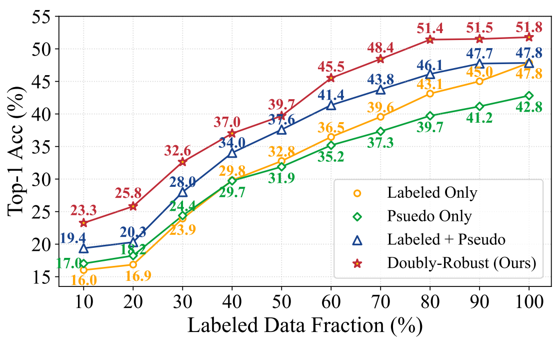
We conduct experiments on both image classification task with ImageNet dataset [91] and 3D object detection task with autonomous driving dataset nuScenes [10].

4.3.1 Image classification

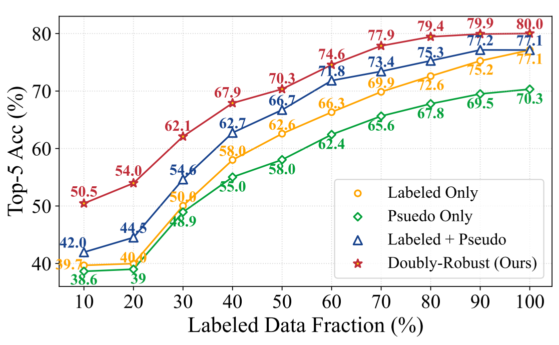
4.3.1.1 Datasets and settings

We evaluate our doubly robust self-training method on the ImageNet100 dataset, which contains a random subset of 100 classes from ImageNet-1k [91], with 120K training images (approximately 1,200 samples per class) and 5,000 validation images (50 samples per class). To further test the effectiveness of our algorithm in a low-data scenario, we create a dataset that we refer to as mini-ImageNet100 by randomly sampling 100 images per class from ImageNet100. Two models were evaluated: (1) DaViT-T [22], a popular vision transformer architecture with state-of-the-art performance on ImageNet, and (2) ResNet50 [30], a classic convolutional network to verify the generality of our algorithm.

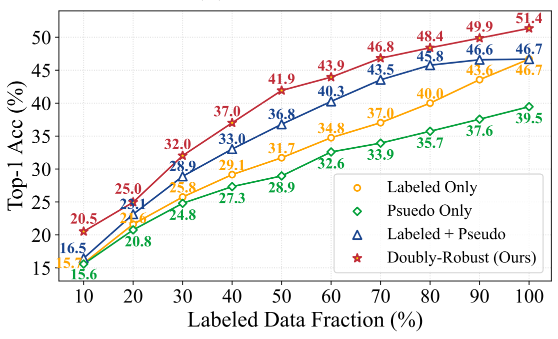
For the training, we use the same data augmentation and regularization strategies following the common practice in [66, 62, 22]. We train all the models with a batch size of 1024 on 8 Tesla V100 GPUs (the batch size is reduced to 64 if the number of training data is less than 1000). We use AdamW [68] optimizer and a simple triangular learning rate schedule [98]. The weight decay is set to 0.05 and the maximal gradient norm is clipped to 1.0. The stochastic depth drop rates are set to 0.1 for all models. During training, we crop images randomly to 224×224, while a center crop is used during evaluation on the validation set. We use a curriculum setting where the α_t grows linearly or quadratically from 0 to 1 throughout the training. To show the effectiveness of our method, we also compare model training with different curriculum learning settings and varying numbers of epochs.



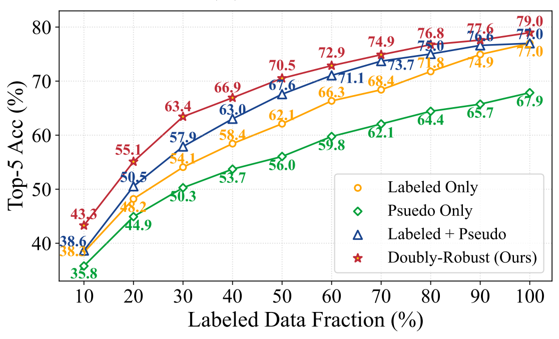
(a) Top-1 on DaViT



(b) Top-5 on DaViT



(c) Top-1 on ResNet50



(d) Top-5 on ResNet50

Fig. 4.3: Comparisons on ImageNet100 using two different network architectures. Both Top-1 and Top-5 accuracies are reported. All models are trained for 20 epochs.

4.3.1.2 Baselines

To provide a comparative evaluation of doubly robust self-training, we establish three baselines: (1) 'Labeled Only' for training on labeled data only (partial training set) with a loss $\mathcal{L}^{\mathsf{TL}}$, (2) 'Pseudo Only' for training with pseudo labels generated for all training samples, and (3) 'Labeled + Pseudo' for a mixture of pseudo-labels and labeled data, with the loss $\mathcal{L}^{\mathsf{SL}}$. See the Appendix for further implementation details and ablations.

4.3.1.3 Results on ImageNet100

We first conduct experiments on ImageNet100 by training the model for 20 epochs using different fractions of labeled data from 1% to 100%. From the results shown in Fig. 4.3, we observe that: (1) Our model outperforms all baseline methods on both two networks by large margins. For example, we achieve 5.5% and 5.3% gains (Top-1 Acc) on DaViT over the 'Labeled + Pseudo' method for 20% and 80% labeled data, respectively. (2) The 'Labeled + Pseudo' method consistently beats the 'Labeled Only' baseline. (3) While 'Pseudo Only' works for smaller fractions of the labeled data (less than 30%) on DaViT, it is inferior to 'Labeled Only' on ResNet50.

4.3.1.4 Results on mini-ImageNet100

We also perform comparisons on mini-ImageNet100 to demonstrate the performance when the total data volume is limited. From the results in Table 4.1, we see our model generally

Labeled Data Percent	Labeled Only		Pseudo Only		Labeled + Pseudo		Doubly robust Loss	
	top1	top5	top1	top5	top1	top5	top1	top5
1	2.72	9.18	**2.81**	9.57	2.73	9.55	2.75	**9.73**
5	3.92	13.34	4.27	13.66	4.27	14.4	**4.89**	**16.38**
10	6.76	20.84	7.27	21.64	7.65	22.48	**8.01**	**21.90**
20	12.3	31.3	13.46	30.79	**13.94**	**32.63**	13.50	32.17
50	20.69	46.86	20.92	45.2	24.9	50.77	**25.31**	**51.61**
80	27.37	55.57	25.57	50.85	30.63	58.85	**30.75**	**59.41**
100	31.07	60.62	28.95	55.35	**34.33**	62.78	34.01	**63.04**

Table 4.1: Comparisons of doubly-robust loss with baselines on mini-ImageNet100, all models trained for 100 epochs.

outperforms all baselines. As the dataset size decreases and the number of training epochs increases, the gain of our algorithm becomes smaller. This is expected, as (1) the models are not adequately trained and thus have noise issues, and (2) there are an insufficient number of ground truth labels to compute the last term of our loss function. In extreme cases, there is only one labeled sample (1%) per class.

4.3.1.5 Ablation Studies

First, we perform ablation studies on varying learning curriculum settings. There are three options for the curriculum setting: 1) $\alpha_t = 1$ throughout the whole training, 2) grows linearly with training iterations $\alpha_t = t/T$, 3) grows quadratically with training iterations $\alpha_t = (t/T)^2$. From results in Table 4.2, we see: the first option achieves comparable performance with the 'Naive Labeled + Pseudo' baseline. Both the linear and quadratic strategies show significant performance improvements: the linear one works better when more labeled data is available, e.g., 70% and 90%, while the quadratic one prefers less labeled data, e.g. 30% and 50%.

Methods	30% GTs		50% GTs		70% GTs		90% GTs	
	top1	top5	top1	top5	top1	top5	top1	top5
Naive Labeled + Pseudo	28.01	54.63	37.6	66.72	43.76	73.42	47.74	77.15
doubly robust, $\alpha_t = 1$	28.43	56.65	38.06	67.18	43.22	73.18	48.52	77.21
doubly robust, $\alpha_t = t/T$ (linear)	30.87	60.98	40.18	71.06	**46.60**	**75.80**	**50.44**	**78.88**
doubly robust, $\alpha_t = (t/T)^2$ (quadratic)	**31.15**	**61.29**	**40.86**	**71.14**	45.50	75.11	49.64	77.77

Table 4.2: Ablation study on different curriculum settings on ImageNet-100. All models are trained in 20 epochs.

Next, we conduct experiments on different training epochs. The results are shown in Table 4.3. Our model is consistently superior to the baselines. And we can observe the gain is larger when the number of training epochs is relatively small, e.g. 20 and 50.

In our original experiments, we mostly focus on a teacher model that is not super accurate, since our method reduces to the original pseudo-labeling when the teacher model is completely correct for all labels. In this experiment, we fully train the teacher model with 300 epochs on ImageNet-100, leading to the accuracy of the teacher model as high as 88.0%. From Figure 4.4, we show that even in this case, our method outperforms the original pseudo-labeling baseline.

Training epochs	Labeled Only		Pseudo Only		Labeled + Pseudo		doubly robust Loss	
	top1	top5	top1	top5	top1	top5	top1	top5
20	16.02	39.68	17.02	38.64	19.38	41.96	**21.88**	**47.18**
50	25.00	51.21	28.90	53.74	30.36	57.04	**36.65**	**65.68**
100	35.57	64.66	44.43	**71.56**	42.44	68.94	**45.98**	70.66

Table 4.3: Ablation study on the number of epochs. All models are trained using 10% labeled data on ImageNet-100.

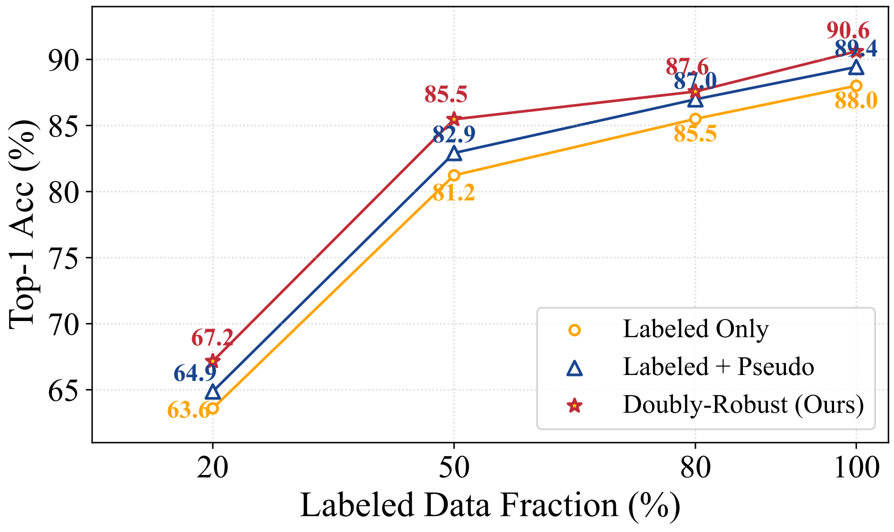


Fig. 4.4: Results on ImageNet-100 using fully trained (300 epochs) DaViT-T with different data fractions.

4.3.1.6 Comparisons with previous SOTAs on CIFAR-10 and CiFAR-100

We compare with another 11 baselines in terms of error rate on CIFAR-10-4K and CIFAR-100-10K under the same settings (i.e., Wide ResNet-28-2 for CIFAR-10 and WRN-28-8 for CIFAR-100). We show that our method is only 0.04 inferior to the best method Meta Pseudo Labels for CIFAR-10-4K, and achieves the best performance for CIFAR-100-10K.

4.3.2 3D object detection

4.3.2.1 Doubly robust object detection

Given a visual representation of a scene, 3D object detection aims to generate a set of 3D bounding box predictions $\{b_i\}_{i\in[m+n]}$ and a set of corresponding class predictions $\{c_i\}_{i\in[m+n]}$. Thus, each single ground-truth annotation $Y_i \in Y$ is a set $Y_i = (b_i, c_i)$ containing a box and a class. During training, the object detector is supervised with a sum of the box regression loss $\mathcal{L}_{loc}$ and the classification loss $\mathcal{L}_{cls}$, i.e. $\mathcal{L}_{obj} = \mathcal{L}_{loc} + \mathcal{L}_{cls}$.

In the self-training protocol for object detection, pseudo-labels for a given scene X_i are selected from the labeler predictions $f(X_i)$ based on some user-defined criteria (typically the model's detection confidence). Unlike in standard classification or regression, Y_i will contain a differing number of labels depending on the number of objects in the

Method	CIFAR-10-4K (error rate, %)	CIFAR-100-10K (error rate, %)
Pseudo-Labeling	16.09	36.21
LGA + VAT	12.06	–
Mean Teacher	9.19	35.83
ICT	7.66	–
SWA	5.00	28.80
MixMatch	4.95	25.88
ReMixMatch	4.72	23.03
EnAET	5.35	–
UDA	4.32	24.50
FixMatch	4.31	23.18
Meta Pesudo Labels	**3.89**	–
Ours	3.93	**22.30**

Table 4.4: Comparisons with previous SOTAs on CiFAR-10 and CIFAR-100.

scene. Furthermore, the number of extracted pseudo-labels $f(X_i)$ will generally not be equal to the number of scene ground-truth labels Y_i due to false positive/negative detections. Therefore it makes sense to express the doubly robust loss function in terms of the individual box labels as opposed to the scene-level labels. We define the doubly robust object detection loss as follows:

$$\mathcal{L}_{obj}^{\mathsf{DR}}(\theta) = \frac{1}{M+N_{ps}} \sum_{i=1}^{M+N_{ps}} \ell_\theta(X_i, f(X_i)) - \frac{1}{N_{ps}} \sum_{i=M+1}^{M+N_{ps}} \ell_\theta(X_i', f(X_i')) + \frac{1}{N} \sum_{i=M+1}^{M+N} \ell_\theta(X_i, Y_i)$$

where M is the total number of pseudo-label boxes from the unlabeled split, N is the total number of labeled boxes, X_i' is the scene with pseudo-label boxes from the *labeled* split, and N_{ps} is the total number of pseudo-label boxes from the *labeled* split. We note that the last two terms now contain summations over a differing number of boxes, a consequence of the discrepancy between the number of manually labeled boxes and pseudo-labeled boxes. Both components of the object detection loss (localization/classification) adopt this form of doubly robust loss.

Labeled Data Fraction	Labeled Only		Labeled + Pseudo		Doubly robust Loss	
	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑
1/24	7.56	18.01	7.60	17.32	**8.18**	**18.33**
1/16	11.15	20.55	11.60	21.03	**12.30**	**22.10**
1/4	25.66	41.41	**28.36**	**43.88**	27.48	43.18

Table 4.5: Performance comparison on nuScenes *val* set.

4.3.2.2 Setting

Our experiments follow the standard approach for semi-supervised detection: we first initialize two detectors, the teacher (i.e., labeler) and the student. First, a random split of varying sizes is selected from the nuScenes training set. We pre-train the teacher network using the ground-truth annotations in this split. Following this, we freeze the weights in the teacher model and then use it to generate pseudo-labels on the entire training set. The student network is then trained on a combination of the pseudo-labels and ground-truth labels originating from the original split. In all of our semi-supervised experiments, we use CenterPoint with a PointPillars backbone as our 3D detection model [127, 48]. The teacher pre-training and student training are both conducted for 10 epochs on 3 NVIDIA RTX A6000 GPUs. We follow the standard nuScenes training setting outlined in [141], with the exception of disabling ground-truth paste augmentation during training to prevent data leakage from the labeled split. To select the pseudo-labels to be used in training the student, we simply filter the teacher predictions by detection confidence, using all detections above a chosen threshold. We use a threshold of 0.3 for all classes, as in [79]. In order to conduct training in a batch-wise manner, we compute the loss over only the samples contained within the batch. We construct each batch to have a consistent ratio of labeled/unlabeled samples to ensure the loss is well-defined for the batch.

4.3.2.3 Main Results

Training Setting	mAP
1400 labeled frames + doubly robust training	11.53
1750 labeled frames + baseline training	11.60

Table 4.6: mAP comparison between two labeled data training settings, illustrating a 20% improvement in data efficiency using the doubly-robust loss with equivalent mAP.

After semi-supervised training, we evaluate our student model performance on the nuScenes *val* set. We compare three settings: training the student model with only the available labeled data (i.e., equivalent to teacher training), training the student model on the combination of labeled/teacher-labeled data using the naive self-training loss, and training the student model on the combination of labeled/teacher-labeled data using our proposed doubly robust loss. We report results for training with 1/24, 1/16, and 1/4 of the total labels in Table 4.5. We find that the doubly robust loss improves both mAP and NDS over using only labeled data and the naive baseline in the lower label regime, whereas performance is slightly degraded when more labels are available. To illustrate the data efficiency gains using our doubly-robust loss, we compare performance between two settings: 1400 labeled frames using doubly-robust training, and 1750 labeled frames using the baseline SSL loss. Shown in Fig. 4.6, we show that using 20% fewer labeled frames, we are able to achieve equivalent performance with our doubly-robust loss to that of the naive loss. Furthermore, we also show a per-class performance breakdown in Table 4.7. We find that the doubly robust loss consistently improves performance for both common (car, pedestrian) and rare classes. Notably, the doubly robust loss is even able to improve upon the teacher in classes for which pseudo-label training *decreases* performance when using the naive training (e.g., barriers and traffic cones).

	Car	Ped	Truck	Bus	Trailer	Barrier	Traffic Cone
Labeled Only	48.6	30.6	8.5	6.2	4.0	6.8	4.4
Labeled + Pseudo	48.8	30.9	8.8	7.5	5.7	6.7	4.0
Improvement	+0.2	+0.3	+0.3	+1.3	**+1.7**	-0.1	-0.4
Doubly robust Loss	51.5	32.9	9.6	8.2	5.2	7.2	4.5
Improvement	**+2.9**	**+2.3**	**+1.1**	**+2.0**	+1.2	**+0.4**	**+0.1**

Table 4.7: Per-class mAP (%) comparison on nuScenes *val* set using 1/16 of total labels in training.

4.3.2.4 Ablation Studies

We compare our approach to another semi-supervised baseline on the 3D object detection task, Pseudo-labeling and confirmation bias [5]. We shows that in multiple settings, our approach surpasses the baseline performance (results shown in Tab. 4.8).

To demonstrate that appropriate quality pseudo-labels are used to train the student detector, we performance ablation experiments varying the detection threshold used to extract pseudo-labels from the teacher model predictions. As shown in Tab. 4.9, we show that training with a threshold of $\tau = 0.3$ outperforms training with a more stringent threshold, and is the appropriate experimental setting for our main experiments.

Table 4.8: Performance comparison with pseudo-labeling baseline on nuScenes *val* set.

Labeled Fraction	Labeled Only		Labeled + Pseudo		Doubly robust Loss		Pseudo-Labeling + Confirmation Bias	
	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑
1/24	7.56	18.01	7.60	17.32	**8.18**	**18.33**	7.80	16.86
1/16	11.15	20.55	11.60	21.03	**12.30**	22.10	12.15	**22.89**

Table 4.9: Doubly Robust Loss performance comparison with differing detection thresholds for pseudo-labels.

Labeled Data Fraction	$\tau = 0.3$				$\tau = 0.5$			
	Labeled+Pseudo		Doubly Robust Loss		Labeled+Pseudo		Doubly Robust Loss	
	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑	mAP↑	NDS↑
1/24	7.56	18.01	**8.18**	**18.33**	7.15	15.82	4.37	13.17
1/16	11.15	20.55	**12.30**	**22.10**	11.05	21.22	8.09	19.70

4.4 Discussion

In this chapter, we propose a novel theoretically-grounded approach to semi-supervised learning, which we dub doubly-robust self-training. In this approach, we replace the standard self-training SSL loss function with a simple re-weighted modification. Theoretically, we analyzed the double-robustness property of the proposed loss, demonstrating its statistical efficiency when the pseudo-labels are accurate. Empirically, we perform experiments on both image classification and 3D object detection on ImageNet and nuScenes, respectively. For both tasks, doubly-robust training outperforms the naive training loss across a variety of labeled data settings.

Although our scheme for doubly-robust self-labeling is simple to implement and effective, several key factors limit its potential use in real-world autolabeling systems. First, we note that the improvement is performance over the baseline autolabeling approach is quite moderate (i.e. 1-2 mAP on 3D object detection), so on its own our improved loss function design cannot fully replicate accuracy achieved using human-labeled data. Another issue arises from the instability encountered during training with the doubly-robust loss. In our experiments, we address this using a curriculum-based loss to help stabilize training, however more work needs to be done to better understand this instability and develop a more robust method for training. A further significant drawback is the assumption underlying our theoretical analysis that the labeled and unlabeled data both arise from the same distribution; in reality, this is very rarely the case. Developing an improved algorithm which is sound in this scenario is another direction needed to make the doubly-robust loss applicable in real self-driving functions.

While further improving semi-supervised learning to aid in large-scale 3D autolabeling still has a ways to come, our combined theoretical and empirical analysis offers an improved understanding of self-labeling that can be easily implemented to offer modest improvement in performance. Doubly-robust self-training pushes the capability of autolabeling forward, aiding in efforts to reduce the significant data labeling costs associated with training 3D perception models for self-driving.

Chapter 5

Prediction Enhanced Autolabeling

In this chapter, we will continue our investigation of data-efficient 3D vision. As opposed to our generalized doubly-robust training approach introduced in Chapter 4, in this chapter we will focus on exploring semi-supervised 3D object detection specifically for autonomous driving, aiming to exploit other parts of the autonomy stack to improve label-efficient perception. We introduce a novel approach, TrajSSL, which leverages motion forecasting models to generate synthetic object tracks which are then used in combination with teacher-generated pseudo-labels. Our approach shows strong performance in experiments, outperforming both the standard SSL baseline, as well as our previously introduced doubly-robust self-training.

5.1 Background

The autonomy stack used to power the “brain" of AVs can be broken up into four key parts: perception, prediction, planning and control. Perception, the main focus of this thesis, allows for the vehicle to sense the world around it. Prediction tries to infer the motion of other dynamic objects in the scene. Planning uses the inferred object trajectories and the goal of the ego vehicle to develop an optimal route through the scene. Control sends raw control signals to the vehicle to enable it’s driving. The four of these are summarized in Fig 5.1. Our previous discussions in this thesis have focused solely on improving perception on its own. Other works have explored the possibility of end-to-end autonomous driving, in which all stages of the autonomy stack are learned simultaneously [31]. However, we consider a different philosophy: leveraging downstream tasks in the autonomy stack to improve perception from a label-efficiency perspective. We note that the prediction pipeline is a form of generative modeling, producing synthetic object trajectories based on previously perceived objects. This synthetic scene data can then be used to enrich semi-supervised training; the following parts of this section will provide more background details.

5.1.1 Semi-Supervised 3D Object Detection

All pseudo-labeling-based approaches to SSL seek to address a key challenge: what is the best strategy for maximizing supervision from high-quality pseudo-labels during training, while minimizing supervision from low-quality ones? In order to address this problem, we first need a quantifiable measure of pseudo-label quality. In the previous chapter, we address this problem from a general perspective, using consistency between pseudo-labels and ground-truth labels on labeled data as a way of quantifying the quality of the teacher model. In the context of object detection, a rudimentary approach is to simply use the teacher model detection confidence score as a proxy for pseudo-label quality. However, particularly for a teacher model trained on a limited dataset, the confidence

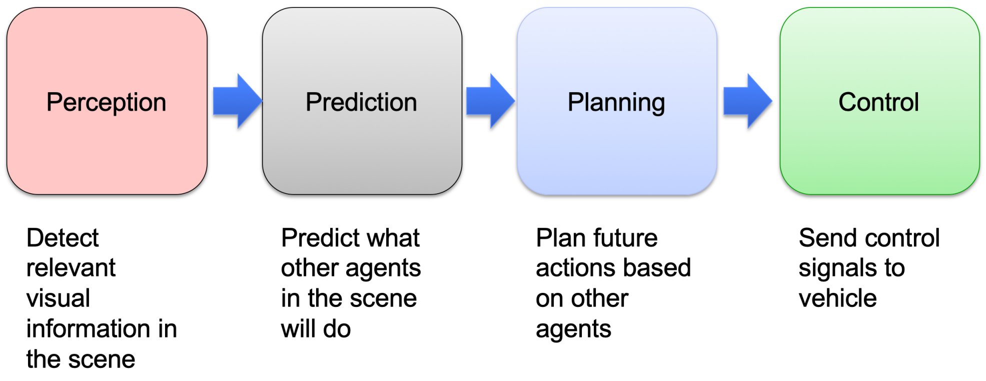

Perception
Prediction
Planning
Control
Detect relevant visual information in the scene
Predict what other agents in the scene will do
Plan future actions based on other agents
Send control signals to vehicle

Fig. 5.1: Visualization of the autonomy stack for AVs including four key functions: perception, prediction, planning and control.

score is often weakly correlated with a pseudo-label's true agreement with a ground truth label [79]. Many other alternative measures have been developed which improve over this rudimentary baseline.

Initial works on semi-supervised object detection primarily focused on the 2D detection task [64, 39, 102, 100, 136, 137]. STAC [100] strongly augments inputs to the student model to enforce augmentation consistency between pseudo-labels. Unbiased teacher [64] uses an exponential moving average (EMA) to update the teacher model during student training. More recent works have also investigated semi-supervised 3D object detection [79, 112, 56, 63, 16, 134]. SESS [134] utilizes three consistency losses to enforce agreement between perturbed variations of the input data. 3DIoUMatch [112] utilizes an IoU estimation module score as a confidence threshold filter. DetMatch [79] takes a multi-modal approach, using agreement between camera model pseudo-labels and LiDAR model pseudo-labels to filter pseudo-labels. HSSDA [63] uses an improved strong data augmentation scheme in combination with hierarchical supervision based on pseudo-label quality to improve training. Playbacks for UDA [129] adopts a temporal refinement of pseudo-labels, using a tracking interpolation/extrapolation module to improve pseudo-label quality in the context of unsupervised domain adaptation.

5.1.2 Temporal 3D Object Detection

In the autonomous driving setting in which object detection is inherently linked to navigating dynamic scenes over time, temporal sequence inputs offer an opportunity for improved detection performance. Several methods for multi-frame 3D object detection have been proposed in the literature [14, 27, 85, 125], ranging in complexity from simple point cloud concatenation [127] to multi-level fusion and refinement schemes [85]. Point concatenation, in which points from multiple frames are directly appended into a single point cloud, is the most common and straightforward approach, but is limited by computation costs associated with increasingly large point clouds. MPPNet [14] and 3DAL [85] use a two-stage refinement where inputs from multiple frames are used to improve bounding box estimates. MoDAR leverages motion forecasting as a vehicle for propagating temporal information, generating virtual points which are added to the point cloud [57]. However, few works have explored leveraging temporal inputs in the context of semi-supervised object

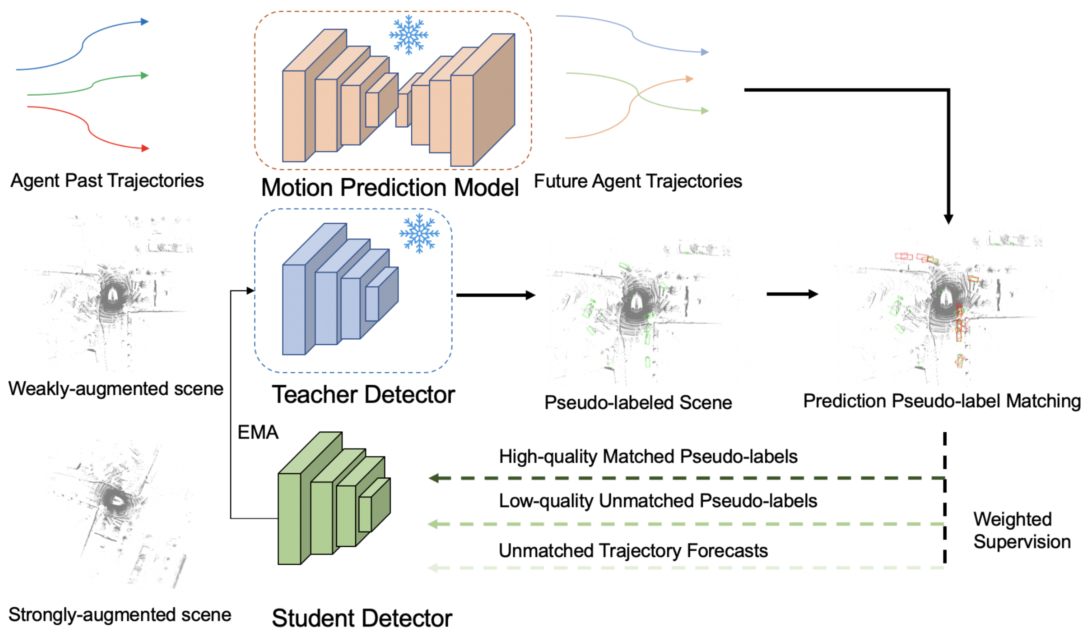


Fig. 5.2: Overview of our proposed method TrajSSL. In addition to a teacher-student SSL framework, we introduce a trajectory prediction model (AgentFormer) which predicts future object trajectories based on past pseudo-label tracks. The inference output of this model is combined with the perception pseudo-labels and an IoU=matching process is performed. Pseudo-labels are then weighted during supervision based on the degree to which they agree with the forecasted trajectories. Meanwhile, predictions which don't match already existing pseudo-labels are added to the training process as down-weighted pseudo-labels.

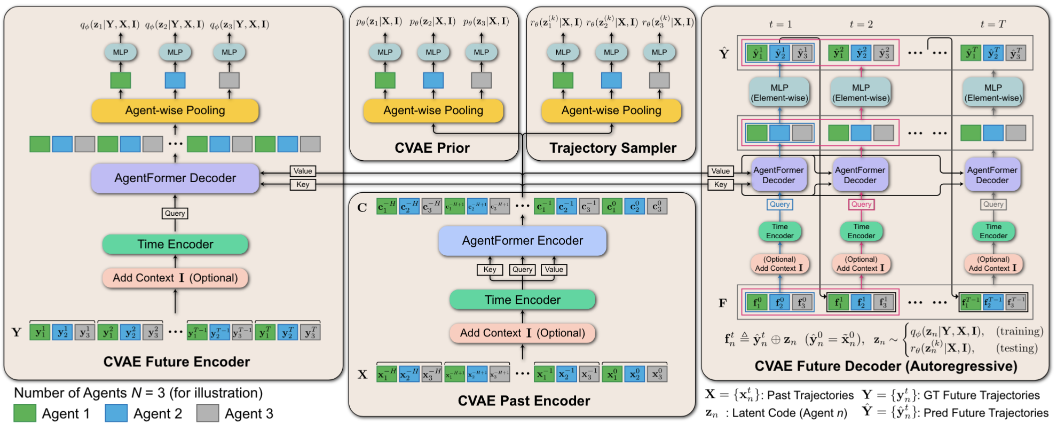
detection.

5.1.3 Motion Prediction

Decision-making in robots/autonomous vehicles navigating dynamic scenes requires an awareness of the motion of other agents in the scene. Trajectory prediction uses the historical motion of other agents in combination with scene-level information (e.g. HD maps) to forecast future agent trajectories. A variety of approaches to exist to trajectory prediction [131, 92, 108, 65, 21], generally relying on neural generative modeling to produce future object trajectories. Agentformer [131] jointly models both temporal and social interactions between agents in the scene, generating trajectories using a conditional variational autoencoder (CVAE) generative model. A few works have also examined training prediction models in a label-efficient manner [120, 11], although this direction remains generally unexplored.

5.2 TrajSSL

In this section, we introduce our proposed approach TrajSSL, and describe in detail both the generation of synthetic trajectories, and the semi-supervised training of a student model leveraging these trajectory outputs. An overview of our approach is shown in Fig. 5.2.



CVAE Future Encoder
CVAE Prior
Trajectory Sampler
CVAE Past Encoder
CVAE Future Decoder (Autoregressive)
Agent-wise Pooling
AgentFormer Decoder
AgentFormer Encoder
Time Encoder
Add Context I (Optional)
MLP
Number of Agents N = 3 (for illustration)
Agent 1
Agent 2
Agent 3
X = {x_n^t}: Past Trajectories
Y = {y_n^t}: GT Future Trajectories
z_n : Latent Code (Agent n)
Ŷ = {ŷ_n^t}: Pred Future Trajectories

Fig. 5.3: Overview of Agentformer architecture. Figure adapted from [131].

5.2.1 Teacher-Student Framework

TrajSSL is built on the frequently-used teacher-student paradigm of SSL. For our experiments, we employ a CenterPoint [127] with PointPillars [48] backbone as our detector models, however any off-the-shelf 3D detector is compatible with this paradigm. First, the teacher model $\mathbf{T}$ is pre-trained on the labeled data samples $\mathcal{D}_l$ until convergence. During student training, the teacher model performs inference on the unlabeled dataset to generate pseudo-labels. The student model $\mathbf{S}$ is then trained on the combination of labeled samples $\{(\mathbf{x}_i^l, \mathbf{y}_i^l)\}_i$ and pseudo-labeled samples $\{(\mathbf{x}_i^u, \mathbf{T}(\mathbf{x}_i^u))\}_i$. During student model training, the teacher detector is improved using an EMA:

$$\theta_\mathbf{T} = \alpha\theta_\mathbf{T} + (1-\alpha)\theta_\mathbf{S} \tag{5.1}$$

where α is the EMA momentum and $\theta_\mathbf{T}$, $\theta_\mathbf{S}$ are the teacher and student model parameters, respectively.

5.2.2 Trajectory Generation

During the teacher pre-training stage, we additionally pre-train a trajectory prediction model for use in the downstream training. For our work, we adopt Agentformer [131] as our motion forecasting model of choice, although our method is compatible with any off-the-shelf model. An overview of Agentformer's architecture is included in Fig. 5.3. Agentformer takes two sets of inputs: a set of agent histories, $\{(\mathbf{x}_i^{-H}, \mathbf{x}_i^{-H+1}, ..., \mathbf{x}_i^0\}_{i=1}^N$ for up to $H+1$ timesteps, and optionally an HD scene-level semantic map. As output, Agentformer generates a set of future trajectory predictions for each input agent, $\{(\mathbf{p}_i^1, \mathbf{p}_i^2, ..., \mathbf{p}_i^T\}_{i=1}^N$ for up to T future timesteps. In this initial stage, Agentformer is pre-trained using the same labeled data split available for semi-supervised training. After completing the pre-training stage, we run teacher model inference on the unlabeled dataset, followed by a multi-object tracker, to generate linked pseudo-label tracks to be used as inputs to Agentformer. Next, we run trajectory prediction inference on all frames of pseudo-labeled scenes, grouping prediction outputs according to their timestamp. Thus, for a sample in the unlabeled set with scene timestamp t, it now has a set of predicted agent

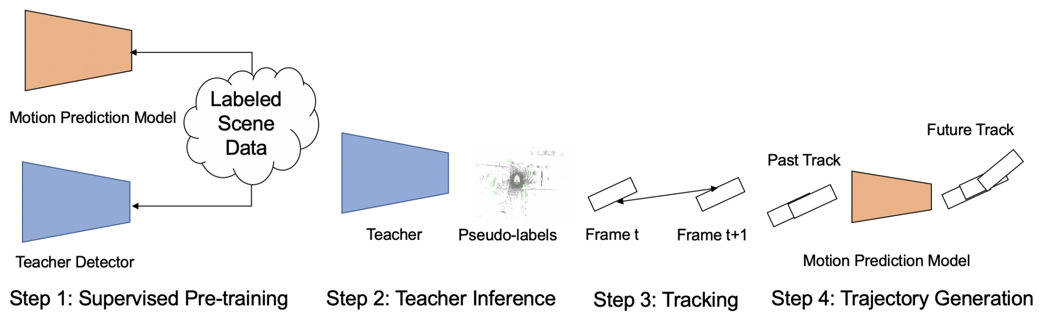


Fig. 5.4: Illustrated process of generation trajectories from pseudo-labels. First, we pre-train both our teacher detector model and our trajectory prediction model using the available labeled scene data. Next, we use the teacher model to run inference on the unlabeled scene data. Next, we link the produced pseudo-labels into tracks of objects across time. Lastly, we feed these tracks into prediction model to generate synthetic trajectories.

locations grouped by prediction context frames: $\{\mathbf{p}_i^{t-T}, \mathbf{p}_i^{t-T+1}, ..., \mathbf{p}_i^{t-1}\}$. A summary of this process is shown in Fig. 5.4.

5.2.3 Matched Prediction Pseudo-label Weighting

After trajectory generation, we now have a set of additional labels to aid in the training of the student detector in addition to the teacher-generated pseudo-labels. The first key insight we exploit is using object forecasts as a measure of *temporal consistency*. If our prediction model predicts a consistent localization for an agent in the scene at a given future timestamp for differing input temporal frames, we argue that this hallucinated object exhibits a strong temporal consistency. Furthermore, if a pseudo-label overlaps with one of these forecasted objects, we can deduce it is likely a higher-quality label, and less likely to be a false positive detection. Thus, by computing the overlap between pseudo-labels and prediction boxes, we have an effective metric for suppressing spurious detections, and emphasizing high-quality labels. To do so, we first compute a maximum IoU between the pseudo-labels and each set of grouped prediction outputs, grouped by context frame. We set a threshold τ_{min_iou} to use for determining whether a pseudo-label and prediction output are successfully "matched". Then, we calculate a per pseudo-label weight based on the number of overlaps meeting the IoU threshold. For the i^{th} pseudo-label, we express this quantitatively as:

$$w_i = \alpha + \sum_{j=t-T}^{t-1} \beta \mathbb{1}\{\max(IoU(\mathbf{x}_i, \{\mathbf{p}|\mathbf{p} \in \mathbf{p}^j\})) \geq \tau_{min\_iou}\} \tag{5.2}$$

where $\mathbb{1}$ is the indicator function and α and β are hyperparameters. The upshot of this weighting scheme is a linear scale for which a greater number of overlapping prediction outputs generates a higher weight. These weights are then used during pseudo-label supervised learning, explained in Sec 5.2.5.

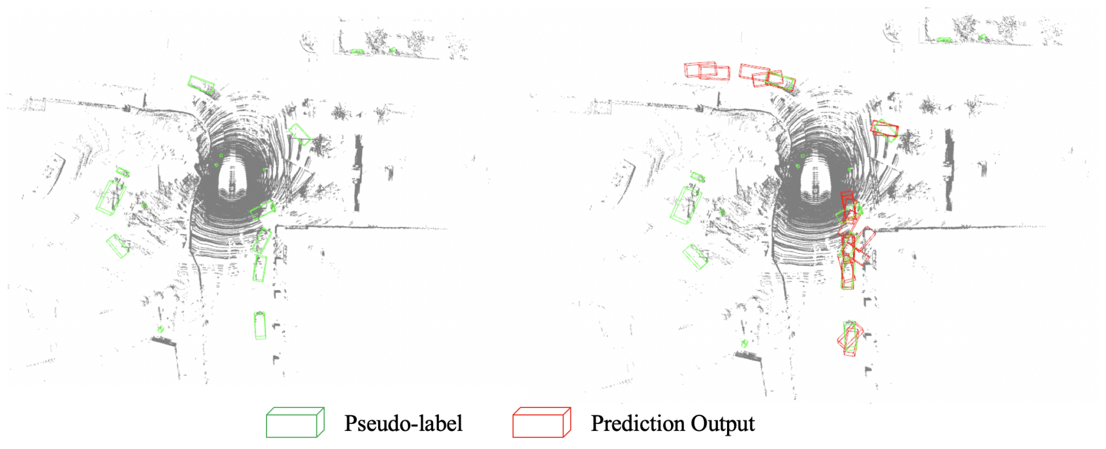


Fig. 5.5: Comparison between a scene containing only teacher-generated pseudo-labels (in green), and the scene augmented with both pseudo-labels and predicted trajectory boxes (in red). Overlapping red and green boxes indicate pseudo-labels exhibiting a high degree of temporal consistency, which are further emphasized during student training. Green boxes without overlap indicate pseudo-labels exhibiting a low degree of temporal consistency, and hence more likely to be a false positive detection. Unmatched red boxes indicate potential missed detections by the teacher model, and are also added as soft targets during training.

5.2.4 Unmatched Prediction-Enhanced Training

While our pseudo-label prediction matching module acts as a filter for pseudo-labels, we also want to be able to correct for the other main source of pseudo-label inaccuracies: false negative (i.e. missed) detections. Our second key insight is in regards to *unmatched* prediction outputs; we note that objects that are missed detections by the teacher model in the current frame, but are successfully tracked in any preceding frames can be recovered based on the forecasted trajectory. Therefore, we propose directly inserting unmatched prediction outputs into the pseudo-label set used during training. To determine unmatched predictions, we once again calculate the maximum IoU between each prediction box and the pseudo-label set. We set a threshold τ_{max_iou}, which is used as the maximum IoU any prediction box can have with a pseudo-label and still be considered "unmatched". We note that in general $\tau_{max\_iou} \neq \tau_{min\_iou}$. While we can directly treat each unmatched detection in a manner equal to a teacher model detection, objects generated by the motion forecasting model are also affected by inaccuracies inherent to predicting future scenes, and thus should not be treated as equivalent to a perceived object. Instead we generate a set of linearly decreasing weights $\gamma_{t-1}, \gamma_{t-2}, ..., \gamma_{t-T}$, where $\gamma_{t-1} \leq 1$, corresponding to a given prediction context frame. We then add each unmatched prediction and assign it the γ value corresponding to the context frame used to generate it. Since our trajectory prediction model becomes less accurate the further in the future it forecasts, we weight unmatched predictions from more recent context frames with greater weight than predictions from further in the past.

5.2.5 Training Objective

During semi-supervised training, we freeze the teacher model weights and only train the student model. We supervise the student model $\mathbf{S}$ with two loss functions: $\mathcal{L}_l$ and $\mathcal{L}_u$, corresponding to the loss on unlabeled and labeled data, respectively.

$$\mathcal{L}_l = \sum_i \mathcal{L}_{reg}(\mathbf{S}(\mathbf{x}_i^l), \mathbf{y}_i^l) + \mathcal{L}_{cls}(\mathbf{S}(\mathbf{x}_i^l), \mathbf{y}_i^l) \tag{5.3}$$

$$\begin{aligned}\mathcal{L}_u = \sum_i \Bigg(& \sum_j w_{ij}\mathcal{L}_{reg}(\mathbf{S}(\mathbf{x}_i^u)_j, \mathbf{T}(\mathbf{x}_i^u)_j) + w_{ij}\mathcal{L}_{cls}(\mathbf{S}(\mathbf{x}_i^u)_j, \mathbf{T}(\mathbf{x}_i^u)_j) \\ & + \sum_k w_{ik}\mathcal{L}_{reg}(\mathbf{S}(\mathbf{x}_i^u)_k, \tilde{p}_{ik}) + w_{ik}\mathcal{L}_{cls}(\mathbf{S}(\mathbf{x}_i^u)_k, \tilde{p}_{ik})\Bigg)\end{aligned} \tag{5.4}$$

where $\mathcal{L}_{cls}$ is the classification loss, $\mathcal{L}_{reg}$ is the bounding box regression loss, w_{ij} is the weight corresponding to the j^{th} pseudo-label of the i^{th} frame, and $\tilde{p}_{ik}$ is the k^{th} unmatched prediction output of the i^{th} frame. During training, we enforce a 1:1 batch ratio of labeled scenes to unlabeled scenes. Thus, the total training objective is defined as simply the sum of the two losses:

$$\mathcal{L}_{tot} = \mathcal{L}_u + \mathcal{L}_l \tag{5.5}$$

5.3 Experimental Results

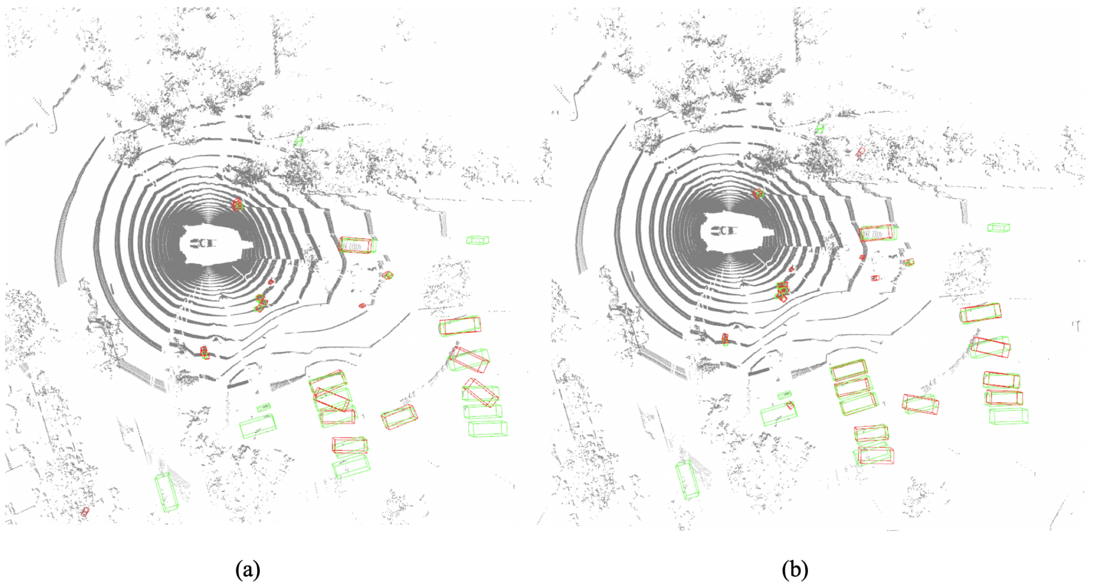


Fig. 5.6: Qualitative results comparison between (a) baseline confidence thresholding SSL approach and (b) TrajSSL. Green boxes indicate ground truth objects, and red indicate model predictions. Visually, we can see that the model trained using TrajSSL performs significantly better, identifying many objects missed by the baseline model, while also generally predicting more accurate bounding box shape and orientation.

To validate our approach, we perform experiments on the nuScenes dataset [10]. Although nuScenes object labels are broken down into ten classes, we restrict our evaluation to the three classes compatible with Agentformer's released models: trucks, cars, and busses. For a comparison baseline, we adopt unbiased teacher [64] with a tuned confidence threshold filtering, which we denote as “confidence thresholding", as similarly proposed in [79].

5.3.1 Implementation Details

We implement our approach using Centerpoint PointPillars as the detection backbones, and Agentformer as our trajectory prediction model. During the pre-training stage, we pre-train both the teacher detection model and Agentformer on the same split of labeled nuScenes training data. For pre-training the detection model, we follow the standard nuScenes training setting outlined in [141], while for pre-training Agentformer we follow the training scheme used in the official implementation [131], with the exception of training DLow [130].

After running teacher model inference on the unlabeled data, we first filter the extracted pseudo-labels with a detection confidence of $\tau_{conf} = 0.3$. To link the extracted pseudo-labels into tracks, we use the greedy tracking algorithm used in [127]. When running AgentFormer inference, we forecast trajectories only for tracks containing at least two frames of past context, while allowing for up to four frames of input. AgentFormer produces up to 12 future frames of trajectory data (6 seconds for 0.5 second keyframe spacing in nuScenes), and we extract predictions on all scene frames for which there is at least a single future frame in the dataset. As AgentFormer only predicts the (x, y) location of an agent in BEV space, we assign the other bounding box attributes of the predicted object according to the attributes of the pseudo-label in the present context frame. Although AgentFormer is capable of producing multiple modes of trajectories for each agent, we extract only the single most probable trajectory for the sake of simplicity.

When matching Agentformer's predictions with teacher model pseudolabels, we set $t_{min\_iou} = 0.3$ as the minimum IoU to count a pseudo-label as being matched to a prediction box. For determining each pseudo-label's weight, we choose $\alpha = 10$ and $\beta = 2$ for the hyperparameters in Eq. 5.2. For a pseudo-label unmatched to any predictions, we maintain its weight at $w = 1$. To determine which prediction boxes remain unmatched, we set $t_{max\_iou} = 0.1$ as the maximum IoU a prediction output can have with an existing pseudo-label. We select the weight parameter to be linearly scaled between $\gamma_{t-1} = 0.75$ and $\gamma_{t-5} = 0.25$.

During the semi-supervised training of the student model, we construct batches with 20 samples, consisting of 10 labeled and 10 unlabeled samples. We conduct training for 10 epochs, using an AdamW optimizer [70] with learning rate 10^{-4}. All training is performed on three NVIDIA A6000 GPUs. As in [79], we use a confidence threshold of $\tau_{conf} = 0.3$. We adopt the same weak-strong augmentation scheme used in 3DIoUMatch [112], in which the data input to the student network is strongly augmented, whereas the data input to the teacher network is weakly augmented. For strong augmentations of the point cloud, we use random flipping, random rotation, and random uniform scaling. For the weak augmentation, we use only random point sub-sampling.

5.3.2 Main Results

We evaluate TrajSSL on the nuScenes dataset for three different labeled data settings: training with 5% labeled data, 10% labeled data, and 20% labeled data. We summarize these results in Tab. 5.1, and additionally include qualitative results for visualization in Fig. 5.6. Across all three settings, TrajSSL improves performance over the confidence thresholding baseline with generally strong performance for all three classes. In the setting with the least labeled data available, we see the most significant performance gains from TrajSSL; in particular, the car and bus classes see an improvement of 1.4 and 4.7 mAP points over the baseline. As the labeled data available increases and the teacher model becomes stronger (hence there exists fewer false positives/negatives to correct for), the relative improvement gained by TrajSSL decreases, though is still noticeable. Additionally, we also compare our approach to doubly-robust training [140], a more general SSL framework. Across all settings and classes, TrajSSL outperforms doubly-robust training. Notably, in the 20% labeled data setting, in which doubly-robust training fails to improve over the confidence thresholding baseline, TrajSSL is still able to gain modest improvements in the bus and truck classes.

Table 5.1: Performance (mAP) comparison on nuScenes validation dataset for car, truck and bus class on a variety of labeled data fraction settings. Our proposed TrajSSL improves performance over previous semi-supervised approaches across all classes in a wide variety of settings. *our re-implementation

Method	5%			10%			20%		
	car	truck	bus	car	truck	bus	car	truck	bus
Labeled Only	49.1	8.7	3.2	61.0	14.2	8.6	66.9	23.0	22.5
SSL Baseline*	52.9	11.2	4.6	63.2	**15.8**	9.9	**70.9**	24.4	27.0
Improvement	+3.8	+2.5	+1.4	+2.2	+1.6	+1.3	+4.0	+1.4	+4.5
Doubly Robust Training*	53.7	11.0	5.9	64.1	14.7	11.0	**70.9**	24.3	26.4
Improvement	+4.6	+2.3	+2.7	+3.1	+ 0.5	+2.4	+4.0	+1.3	+3.9
Ours	**54.3**	**11.4**	**9.3**	**64.7**	15.7	**11.9**	70.1	**24.8**	**27.5**
Improvement	+5.2	+2.7	+6.1	+3.7	+1.5	+3.3	+3.2	+1.8	+5.0

5.3.3 Ablation Studies

In this section, we perform ablation studies on the various aspects of our TrajSSL framework. We perform all ablation experiments using the 5% labeled training data setting.

5.3.3.1 False Positive/Negative Compensation

The first set of ablation experiments we perform is to verify the improvement gained from our two strategies for suppressing false positives and directly correcting false negatives. We summarize the results of these experiments in Tab. 5.2. We find the most significant improvement arises from the up-weighting of pseudo-labels which are matched to a prediction output; while the improvement to the truck class is modest, the bus and car class see an improvement of +4.3 mAP and +1.2 mAP, respectively. This supports our hypothesis of temporal consistency established through trajectory forecasts being a good metric for pseudo-label quality.

Our second key component, direct addition of prediction outputs to correct false negatives, results in a further modest increase in performance, improving the car and bus class by +0.2 mAP and +0.4 mAP, respectively while truck class mAP remains unchanged. While the ability to directly replace false negatives with forecasted objects is limited by the quality of the pseudo-label tracks used as input to Agentformer, nonetheless a consistent improvement verifies that unmatched prediction objects contain useful information gained from temporal context and can improve the student model training.

Table 5.2: Ablation of two main strategies of TrajSSL.

	Car	Truck	Bus
Labeled Only	49.1	8.7	3.2
+ Teacher-Student SSL	52.9	11.2	4.6
+ Matched Prediction Pseudo-label Weighting	54.1	11.4	8.9
+ Unmatched Prediction Addition	**54.3**	**11.4**	**9.3**

5.3.3.2 Trajectory Time Horizon

Table 5.3: AgentFormer prediction accuracy for varying future frame timestamps.

	ADE (m)	FDE (m)
10 Frame Predictions	3.31	6.26
5 Frame Predictions	1.68	2.66
2 Frame Predictions	0.90	1.11

The next key aspect of our approach we want to verify is the utility of Agentformer's future predictions, especially in this setting in which it is trained on limited data. To do so, we perform experiments using a varying number of temporal frame outputs from Agentformer, which is capable of predicting up to 12 frames (6 seconds in the context of nuScenes) into the future. We include the results in Tab. 5.4. We see that adopting TrajSSL for even one single frame of trajectory outputs significantly improves performance over the non-temporal baseline. Increasing the number of Agentformer output frames to 5 frames results in a further increase in mAP, although the improvement is far less dramatic then the jump from one to two frames, due to the declining accuracy of trajectory prediction models as they predict further into the future. Going further to 8 or 10 frames degrades performance from using 5 frames for both the car and bus class, while slightly improving the truck class by +0.1 mAP, indicating forecasted objects this far into the future aren't accurate enough to successfully integrate into TrajSSL. Hence we select 5 frames as the optimum used in our main experiments.

In addition to our direct ablation studies on the number of predictions frames used in TrajSSL, we also quantitatively evaluate Agentformer's performance when pre-trained on 5% of the nuScenes training set. We use two metrics to evaluate prediction accuracy: average displacement error (ADE) and final displacement error (FDE). Average displacement error is an average of the displacement between the prediction and the ground truth object across the entire trajectory, whereas the final displacement error is simply the displacement for the furthest future prediction. We summarize these results in Tab. 5.3. For 10 frames of future predictions, we the FDE is 6.26 meters, larger than the average vehicle size, indicating that bounding boxes output by AgentFormer are unlikely to be succesfully matched to an existing object/pseudo-label. However, for 5 frames of future predictions, the FDE is a more reasonable 2.66 meters, roughly on the scale of the size of an average vehicle and thus more useful in our SSL scheme.

Table 5.4: Ablation of number of prediction frames used in TrajSSL.

	Car	Truck	Bus
+1 Frame (SSL Baseline)	52.9	11.2	4.6
+2 Frames	53.9	11.0	8.5
+5 Frames	**54.3**	11.4	**9.3**
+8 Frames	53.8	**11.5**	8.7
+10 Frames	53.9	**11.5**	8.8

5.3.3.3 Linear Extrapolation Baseline Comparison

A further ablation study we perform is to directly probe the necessity of a complex neural model (such as Agentformer) for generating the future forecasts of scene objects. As a baseline, we consider performing a linear extrapolation using the model-predicted velocity of each object to predict future object locations, after which we use our already proposed weighting mechanism. We compare these two approaches in Tab. 5.5. Using the linear extrapolation approach is still able to improve the SSL baseline on both the car and bus class. However, across all three classes, predicting future trajectories using Agentformer noticeably outperforms the simple linear extrapolation approach. We attribute this to the fact that a) the teacher model (particularly when pre-trained on limited data) is poor at predicting velocity accurately, making linear extrapolation less accurate and b) particularly for longer time-horizon forecasting, linear extrapolation is too simple to capture the complex scene dynamics to accurately predict agent trajectories. Thus, a powerful trajectory prediction model, even when trained on a sparse dataset, is a key ingredient to maximizing the effectiveness of TrajSSL.

Table 5.5: Comparison of our approach using Agentformer versus using a linear extrapolation.

	Car	Truck	Bus
SSL Baseline	52.9	11.2	4.6
Prediction Model (AgentFormer)	**54.3**	**11.4**	**9.3**
Linear Extrapolation	53.2	11.0	8.4

5.3.3.4 Weighting Hyperparameters

Table 5.6: Impact of weighting parameters α, β.

	Car	Truck	Bus
$\alpha = 2$, $\beta = 0$	53.5	**11.4**	7.2
$\alpha = 5$, $\beta = 0$	53.5	11.2	7.8
$\alpha = 10$, $\beta = 0$	53.9	11.0	8.5
$\alpha = 10$, $\beta = 2$	**54.3**	**11.4**	**9.3**
$\alpha = 20$, $\beta = 2$	53.6	10.9	8.2

To determine the optimal values of various hyperparameters associated with assigning pseudo-label weight values, we perform a series of experiments. First, we consider the weighting hyperparameters associated with pseudo-labels matched to predictions, α and

β. We include the experimental results in Tab. 5.6. We first perform experiments with $\beta = 0$, i.e. a weighting scheme in which any pseudo-label matched to any number of prediction boxes gets the same weight. However, we find that introducing a non-zero β, i.e. increasing a pseudo-label's weight as agreement with more temporal context frame predictions increases, further improves upon the static weighting scheme. Thus we adopt this weighting scheme with $\alpha = 10$, $\beta = 2$ into our final model.

Table 5.7: Impact of weighting parameter γ.

	Car	Truck	Bus
$\gamma = 0$	54.1	11.4	8.9
$\gamma = 1$	53.4	10.7	4.8
$\gamma = 0.5$	53.9	**11.5**	9.2
$\gamma = 0.25$	**54.3**	11.3	9.1
$\gamma_{t-1} = 0.75$, $\gamma_{t-5} = 0.25$	**54.3**	11.4	9.3
$\gamma_{t-1} = 1.0$, $\gamma_{t-5} = 0.5$	54.2	11.3	**9.4**

The other weighting hyperparameter we consider is the value of weights assigned to unmatched prediction boxes added during training, previously introduced as γ. We summarize these experiments in Tab. 5.7. First, we consider simply choosing $\gamma = 1$, meaning we simply treat added prediction boxes as identical to teacher pseudo-labels. However, we find this approach decreases mAP relative to simply not adding any of the prediction boxes. Setting $\gamma < 1$, such as $\gamma = 0.5$ or $\gamma = 0.25$, is able to slightly increase mAP over the baseline of not adding any prediction boxes. However, we find that a variable weighting scheme, in which γ is scaled based on the nearest-in-time context frame used to generate the trajectory, is able to further improve our results. We choose a linearly-scaled weighting mechanism, in which $\gamma_{t-1} = 0.75$ and $\gamma_{t-5} = 0.25$, for our final choise of hyperparameters.

5.3.3.5 Matching Threshold

Table 5.8: Ablation experiments on τ_{min_iou}.

	Car	Truck	Bus
$\tau_{min\_iou} = 0.3$	**54.3**	**11.4**	**9.3**
$\tau_{min\_iou} = 0.5$	52.8	10.6	4.8

Another set of ablations we perform is on the IoU threshold τ_{min_iou} between pseudo-labels and prediction outputs when determining which pseudo-labels to up-weight. We include the results in Tab. 5.8. We find that increasing the IoU threshold to 0.5 decreases performance significantly, thus we set $\tau_{min\_iou} = 0.3$ in all our experiments.

5.4 Discussion

In this chapter, we proposed a novel framework for semi-supervised 3D object detection in autonomous driving scenarios based on leveraging trajectory prediction models to enhance pseudo-label training, which we dub TrajSSL. TrajSSL uses outputs from Agentformer, a trajectory forecasting model, to enhance the training of the student detector in two key ways: first, it uses these predicted objects to locate higher-quality pseudo-labels and up-weight them during the training process. Second, unmatched outputs are used to directly compensate for missed detections. On experiments using the nuScenes dataset, TrajSSL outperforms previous SSL approaches in a wide variety of settings, demonstrating the success of our prediction-enhanced approach to SSL and opening the possibility toward more creative ways of leveraging the autonomy stack to improve perception data efficiency.

Although TrajSSL is able to improve upon the state-of-the-art, it still has several limitations. TrajSSL is fundamentally limited by the challenge of training a trajectory forecasting model using the limited data available in a semi-supervised setting. Label-efficient training of these forecasting models are still relatively unexplored, and offer the potential to improve the performance of our method. Additionally, our implementation of TrajSSL only operates on vehicles of a similar class due to the challenge of modeling the paths of agents with vastly different velocity scales. Expanding TrajSSL to function on the other dynamic objects in driving scenes, such as pedestrians, is another key future direction.

In addition to improving upon previously introduced SSL techniques for 3D object detection, TrajSSL also takes a novel philosophy to data-efficient perception through leveraging the AV stack in an unconventional manner. This work serves as a promising starting point toward leveraging the power of motion forecasting as not only a critical function in aiding self-driving planners, but also as a tool to improve AV perception.

Chapter **6**

Conclusion

In this dissertation, we introduce our multi-faceted work toward developing efficient vision for self-driving cars. We consider efficiency as defined by four metrics: accuracy, latency, compute, and data. In Chapter 2, we consider tackling perception from a hardware and latency angle, and introduce a novel form of optoelectronic reservoir computing which combines a delay-based reservoir computer with pre-processing through a set of untrained convolutional layers. With limited hardware, our implementation can reduce training time by up to a factor of 10x when compared to training a standard CNN model using digital hardware. In Chapter 3, we introduce our novel algorithm for LiDAR-camera fusion-based 3D object detection, CFF, which aims to address efficiency from the latency perspective. Through leveraging a sparse feature fusion scheme projecting only a set of key camera pixels to birds-eye-view, CFF achieves accuracy competitive with other fusion approaches while significantly reducing runtime latency. In Chapters 4 and 5, we address the problem of data efficiency, aiming to improve semi-supervised learning for 3D object detection. In Chapter 4, we introduce a novel theoretically-grounded approach to semi-supervised learning, which we dub doubly-robust self-training. Through a simple modification of the standard self-training loss function, we deliver a self-training method more theoretically sound across scenarios of both low-quality pseudo-labels and high-quality pseudo-labels. Empirically, we evaluate our method for both image classification and 3D object detection and are able to achieve superior performance to the self-labeling baseline. Lastly, in Chapter 5, we introduce another approach to semi-supervised 3D object detection, TrajSSL, which leverages the motion prediction part of the autonomy stack to improve perception data efficiency. Through using prediction model outputs to establish temporal consistency and correct for both false positive and false negative detections, TrajSSL improves upon the state-of-the-art for semi-supervised 3D object detection with a highly creative approach with significant potential for future improvement. Although the goal of full self-driving still remains elusive, our work paves the way for novel outside the box approaches to realizing efficient self-driving cars.

Bibliography

[1] Alata, R., Pauwels, J., Van der Sande, G., Bouwens, A., Haelterman, M., Massar, S.: Phase noise robustness of a coherent spatially parallel optical reservoir. In: 2019 Conference on Lasers and Electro-Optics Europe European Quantum Electronics Conference (CLEO/Europe-EQEC). pp. 1–1 (2019). https://doi.org/10.1109/CLEOE-EQEC.2019.8872207

[2] Angelopoulos, A.N., Bates, S., Fannjiang, C., Jordan, M.I., Zrnic, T.: Prediction-powered inference. arXiv preprint arXiv:2301.09633 (2023)

[3] Antonik, P., Marsal, N., Rontani, D.: Large-scale spatiotemporal photonic reservoir computer for image classification. IEEE Journal of Selected Topics in Quantum Electronics **26**(1), 1–12 (2020)

[4] Appeltant, L., Soriano, M.C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C.R., Fischer, I.: Information processing using a single dynamical node as complex system. Nature Communications **2**, 466–468 (2011)

[5] Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Pseudo-labeling and confirmation bias in deep semi-supervised learning. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE (2020)

[6] Bai, X., Hu, Z., Zhu, X., Huang, Q., Chen, Y., Fu, H., Tai, C.L.: TransFusion: Robust Lidar-Camera Fusion for 3d Object Detection with Transformers. CVPR (2022)

[7] Berthelot, D., Carlini, N., Cubuk, E.D., Kurakin, A., Sohn, K., Zhang, H., Raffel, C.: Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. arXiv preprint arXiv:1911.09785 (2019)

[8] Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.: Mixmatch: A holistic approach to semi-supervised learning. arXiv preprint arXiv:1905.02249 (2019)

[9] Bibby, J.M., Mardia, K.V., Kent, J.T.: Multivariate Analysis. Academic Press (1979)

[10] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019)

[11] Chen, G., Chen, Z., Fan, S., Zhang, K.: Unsupervised sampling promoting for stochastic human trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17874–17884 (2023)

[12] Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6526–6534. IEEE Computer Society, Los Alamitos, CA, USA (jul 2017). https://doi.org/10.1109/CVPR.2017.691, `https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.691`

[13] Chen, X., Zhang, T., Wang, Y., Wang, Y., Zhao, H.: Futr3d: A unified sensor fusion framework for 3d detection. arXiv preprint arXiv:2203.10642 (2022)

[14] Chen, X., Shi, S., Zhu, B., Cheung, K.C., Xu, H., Li, H.: Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection. In: Computer Vision – ECCV 2022: 17th European Conference. pp. 680–697 (2022)

[15] Chen, Z., Li, Z., Zhang, S., Fang, L., Jiang, Q., Zhao, F., Zhou, B., Zhao, H.: Autoalign: Pixel-instance feature aggregation for multi-modal 3d object detection. arXiv preprint arXiv:2201.06493 (2022)

[16] Chen, Z., Jing, L., Yang, L., Li, Y., Li, B.: Class-level confidence based 3d semi-supervised learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 633–642 (January 2023)

[17] Chollet, F.: Keras documentation: Simple mnist convnet (Jun 2015), https://keras.io/examples/vision/mnist_convnet/

[18] Cireşan, D., Meier, U., Gambardella, L., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. Neural computing **22**(12), 3207–3220. (2010)

[19] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 764–773 (2017). https://doi.org/10.1109/ICCV.2017.89

[20] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence **44**(7), 3366–3385 (2021)

[21] Deo, N., Wolff, E., Beijbom, O.: Multimodal trajectory prediction conditioned on lane-graph traversals. In: 5th Annual Conference on Robot Learning (2021)

[22] Ding, M., Xiao, B., Codella, N., Luo, P., Wang, J., Yuan, L.: Davit: Dual attention vision transformers. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV. pp. 74–92. Springer (2022)

[23] Eldesokey, A., Felsberg, M., Khan, F.S.: Confidence propagation through cnns for guided sparse depth regression. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(10), 2423–2436 (2020). https://doi.org/10.1109/TPAMI.2019.2929170

[24] Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**, 1789–1819 (2021)

[25] Hafizovic, S., Heer, F., Ugniwenko, T., Frey, U., Blau, A., Ziegler, C., Hierlemann, A.: A cmos-based microelectrode array for interaction with neuronal cultures. Journal of Neuroscience Methods **164**(1), 93–106 (2007). https://doi.org/https://doi.org/10.1016/j.jneumeth.2007.04.006, https://www.sciencedirect.com/science/article/pii/S0165027007001781

[26] Harkhoe, K., Verschaffelt, G., Katumba, A., Bienstman, P., Van der Sande, G.: Demonstrating delay-based reservoir computing using a compact photonic integrated chip. Opt. Express **28**(3), 3086–3096 (Feb 2020). https://doi.org/10.1364/OE.382556, http://www.opticsexpress.org/abstract.cfm?URI=oe-28-3-3086

[27] He, C., Li, R., Zhang, Y., Li, S., Zhang, L.: Msf: Motion-guided sequential fusion for efficient 3d object detection from point cloud sequences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5196–5205 (June 2023)

[28] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2980–2988 (2017). https://doi.org/10.1109/ICCV.2017.322

[29] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

[30] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)

[31] Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., Lu, L., Jia, X., Liu, Q., Dai, J., Qiao, Y., Li, H.: Planning-oriented autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023)

[32] Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing **70**(1), 489–501 (2006). https://doi.org/https://doi.org/10.1016/j.neucom.2005.12.126

[33] Jacobson, P., Shirao, M., Yu, K., Su, G.L., Wu, M.C.: Image classification using delay-based optoelectronic reservoir computing. In: Jalali, B., Kitayama, K. (eds.) AI and Optical Data Sciences II. vol. 11703, pp. 120 – 126. International Society for Optics and Photonics, SPIE (2021). https://doi.org/10.1117/12.2578062, https://doi.org/10.1117/12.2578062

[34] Jacobson, P., Shirao, M., Yu, K., Su, G.L., Wu, M.C.: Hybrid convolutional optoelectronic reservoir computing for image recognition. Journal of Lightwave Technology **40**(3), 692–699 (2022). https://doi.org/10.1109/JLT.2021.3124520

[35] Jacobson, P., Zhou, Y., Zhan, W., Tomizuka, M., Wu, M.C.: Center feature fusion: Selective multi-sensor fusion of center-based objects. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 8312–8318 (2023). https://doi.org/10.1109/ICRA48891.2023.10160616

[36] Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science **304**, 78–80 (2004)

[37] Jaeger, H.: The “echo state” approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)

[38] Jaritz, M., Charette, R.D., Wirbel, E., Perrotton, X., Nashashibi, F.: Sparse and dense data with cnns: Depth completion and semantic segmentation. In: 2018 International Conference on 3D Vision (3DV). pp. 52–60 (2018). https://doi.org/10.1109/3DV.2018.00017

[39] Jeong, J., Lee, S., Kim, J., Kwak, N.: Consistency-based semi-supervised learning for object detection. In: Advances in Neural Information Processing Systems (2019)

[40] Jeyachandran, S.: Introducing the 5th-generation waymo driver: Informed by experience, designed for scale, engineered to tackle more environments. https://waymo.com/blog/2020/03/introducing-5th-generation-waymo-driver/ (2020)

[41] Jiang, C.M., Najibi, M., Qi, C.R., Zhou, Y., Anguelov, D.: Improving the intra-class long-tail in 3d detection via rare example mining. In: Computer Vision–ECCV 2022: 17th European Conference. pp. 158–175. Springer (2022)

[42] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3992–4003 (2023). https://doi.org/10.1109/ICCV51070.2023.00371

[43] Krizhevsky, A.: Learning multiple layers of features from tiny images pp. 32–33 (2009), https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[44] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems (NeurIPS). pp. 1097–1105 (2012)

[45] Ku, J., Harakeh, A., Waslander, S.L.: In defense of classical image processing: Fast depth completion on the cpu. In: 2018 15th Conference on Computer and Robot Vision (CRV). pp. 16–22. IEEE (2018)

[46] Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.: Joint 3d proposal generation and object detection from view aggregation. IROS (2018)

[47] Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=BJ6oOfqge

[48] Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 12689–12697 (2019)

[49] Larger, L., Baylón-Fuentes, A., Martinenghi, R., Udaltsov, V.S., Chembo, Y.K., Jacquot, M.: High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. Physical Review X **17**, 011015 (2017)

[50] Larger, L., Soriano, M.C., Brunner, D., Appeltant, L., Gutierrez, J.M., Pesquera, L., Mirasso, C.R., Fischer, I.: Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. Optics Express **20**(3), 3241 (2012)

[51] Lassig, R., Lorenz, M., Sissimatos, E., Wicker, I., Buchner, T.: Robotics outlook 2030: How intelligence and mobility will shape the future. https://www.bcg.com/publications/2021/how-intelligence-and-mobility-will-shape-the-future-of-the-robotics-industry (2021)

[52] LeCun, Y.: The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/ https://ci.nii.ac.jp/naid/10027939599/en/

[53] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[54] Lee, D.H.: Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. ICML 2013 Workshop : Challenges in Representation Learning (WREPL) (07 2013)

[55] Lee, R.H., Mulder, E.B., Hopkins, J.B.: Mechanical neural networks: Architected materials that learn behaviors. Science Robotics **7** (2022)

[56] Li, J., Liu, Z., Hou, J., Liang, D.: Dds3d: Dense pseudo-labels with dynamic threshold for semi-supervised 3d object detection. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2023)

[57] Li, Y., Qi, C.R., Zhou, Y., Liu, C., Anguelov, D.: Modar: Using motion forecasting for 3d object detection in point cloud sequences. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9329–9339 (2023)

[58] Li, Y., Yu, A.W., Meng, T., Caine, B., Ngiam, J., Peng, D., Shen, J., Wu, B., Lu, Y., Zhou, D., Le, Q.V., Yuille, A., Tan, M.: Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In: CVPR (2022)

[59] Li, Y., Bao, H., Ge, Z., Yang, J., Sun, J., Li, Z.: Bevstereo: Enhancing depth estimation in multi-view 3d object detection with temporal stereo. Proceedings of the AAAI Conference on Artificial Intelligence **37**(2), 1486–1494 (Jun 2023). https://doi.org/10.1609/aaai.v37i2.25234, https://ojs.aaai.org/index.php/AAAI/article/view/25234

[60] Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. arXiv preprint arXiv:2203.17270 (2022)

[61] Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3d object detection. In: ECCV (2018)

[62] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. pp. 2980–2988 (2017)

[63] Liu, C., Gao, C., Liu, F., Li, P., Meng, D., Gao, X.: Hierarchical supervision and shuffle data augmentation for 3d semi-supervised object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023)

[64] Liu, Y.C., Ma, C.Y., He, Z., Kuo, C.W., Chen, K., Zhang, P., Wu, B., Kira, Z., Vajda, P.: Unbiased teacher for semi-supervised object detection. In: Proceedings of the International Conference on Learning Representations (ICLR) (2021)

[65] Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal motion prediction with stacked transformers. Computer Vision and Pattern Recognition (2021)

[66] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows (2021)

[67] Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D., Han, S.: Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. arXiv (2022)

[68] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)

[69] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)

[70] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)

[71] Maass, W., Natschlager, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation **14**(11), 2531–2560 (2002)

[72] Maksymov, I.S.: Analogue and physical reservoir computing using water waves: Applications in power engineering and beyond. Energies **16**(14) (2023). https://doi.org/10.3390/en16145366, https://www.mdpi.com/1996-1073/16/14/5366

[73] Marshall, A.W., Olkin, I.: Multivariate chebyshev inequalities. The Annals of Mathematical Statistics pp. 1001–1014 (1960)

[74] Mousavian, A., Anguelov, D., Flynn, J., Košecká, J.: 3d bounding box estimation using deep learning and geometry. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5632–5640 (2017). https://doi.org/10.1109/CVPR.2017.597

[75] Nakajima, K., Hauser, H., Kang, R., Guglielmino, E., Caldwell, D., Pfeifer, R.: A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. Frontiers in Computational Neuroscience **7** (2013). https://doi.org/10.3389/fncom.2013.00091, https://www.frontiersin.org/articles/10.3389/fncom.2013.00091

[76] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)

[77] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)

[78] Paquot, Y., Duport, F., Smerieri, A., Dambre, J., Schrauwen, B., Haelterman, M., Massar, S.: Optoelectronic reservoir computing. Scientific Reports **2**, 1–6 (2012)

[79] Park, J., Xu, C., Zhou, Y., Tomizuka, M., Zhan, W.: Detmatch: Two teachers are better than one for joint 2D and 3D semi-supervised object detection. Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel (2022)

[80] Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: Proceedings of the European Conference on Computer Vision (2020)

[81] Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision (2019)

[82] Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. arXiv preprint arXiv:1711.08488 (2017)

[83] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2016)

[84] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)

[85] Qi, C.R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., Anguelov, D.: Offboard 3D object detection from point cloud sequences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021)

[86] Qian, R., Garg, D., Wang, Y., You, Y., Belongie, S., Hariharan, B., Campbell, M., Weinberger, K.Q., Chao, W.L.: End-to-end pseudo-lidar for image-based 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5881–5890 (2020)

[87] Ravindran, R., Santora, M.J., Jamali, M.M.: Multi-object detection and tracking, based on dnn, for autonomous vehicles: A review. IEEE Sensors Journal **21**(5), 5668–5677 (2021). https://doi.org/10.1109/JSEN.2020.3041615

[88] Reading, C., Harakeh, A., Chae, J., Waslander, S.L.: Categorical depth distribution network for monocular 3d object detection. CVPR (2021)

[89] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28. Curran Associates, Inc. (2015), https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[90] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10674–10685 (2022). https://doi.org/10.1109/CVPR52688.2022.01042

[91] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision **115**, 211–252 (2015)

[92] Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: Computer Vision – ECCV 2020: 16th European Conference (2020)

[93] Shen, Y., Harris, N.C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., Sun, X., Zhao, S., Larochelle, H., Englund, D., Solja, M.: Deep learning with coherent nanophotonic circuits. Nature Photonics **11**, 441–447 (2017)

[94] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10526–10535. IEEE Computer Society, Los Alamitos, CA, USA (jun 2020). https://doi.org/10.1109/CVPR42600.2020.01054, https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01054

[95] Shin, K., Kwon, Y.P., Tomizuka, M.: Roarnet: A robust 3d object detection based on region approximation refinement. In: 2019 IEEE Intelligent Vehicles Symposium (IV). pp. 2510–2515 (2019). https://doi.org/10.1109/IVS.2019.8813895

[96] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), https://api.semanticscholar.org/CorpusID:14124313

[97] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR) (2015)

[98] Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications. vol. 11006, p. 1100612. International Society for Optics and Photonics (2019)

[99] Sohn, K., Berthelot, D., Li, C.L., Zhang, Z., Carlini, N., Cubuk, E.D., Kurakin, A., Zhang, H., Raffel, C.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. arXiv preprint arXiv:2001.07685 (2020)

[100] Sohn, K., Zhang, Z., Li, C.L., Zhang, H., Lee, C.Y., Pfister, T.: A simple semi-supervised learning framework for object detection. In: arXiv:2005.04757 (2020)

[101] Takano, K., Sugano, C., Inubushi, M., Yoshimura, K., Sunada, S., Kanno, K., Uchida, A.: Compact reservoir computing with a photonic integrated circuit. Optics Express **26**(22), 29424–29439 (2018)

[102] Tang, Y., Chen, W., Luo, Y., Zhang, Y.: Humble teachers teach better students for semi-supervised object detection. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 3131–3140 (2021)

[103] Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/68053af2923e00204c3ca7c6a3150cf7-Paper.pdf

[104] Tikhonov, A.N., Goncharsky, A.V., Stepanov, V.V., Yagola, A.G.: Numerical Methods for the Solution of Ill-Posed Problems, vol. 328. Springer, New York, NY, USA (1995)

[105] Tingting Liang, Hongwei Xie, K.Y.Z.X.Z.L.Y.W.T.T.B.W., Tang, Z.: BEVFusion: A Simple and Robust LiDAR-Camera Fusion Framework. In: Neural Information Processing Systems (NeurIPS) (2022)

[106] Van Gansbeke, W., Neven, D., De Brabandere, B., Van Gool, L.: Sparse and noisy lidar completion with rgb guidance and uncertainty. In: 2019 16th International Conference on Machine Vision Applications (MVA). pp. 1–6 (2019). https://doi.org/10.23919/MVA.2019.8757939

[107] Vandoorne, K., Mechet, P., Vaerenbergh, T.V., Fiers, M., Morthier, G., Verstraeten, D., Schrauwen, B., Dambre, J., Bienstman, P.: Experimental demonstration of reservoir computing on a silicon photonics chip. Nature Communications **5**, 3541 (2014)

[108] Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K.S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C.P., Anguelov, D., Sapp, B.: Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 7814–7821 (2022)

[109] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[110] Vora, S., Lang, A.H., Helou, B., Beijbom, O.: Pointpainting: Sequential fusion for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)

[111] Wang, C., Ma, C., Zhu, M., Yang, X.: Pointaugmenting: Cross-modal augmentation for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11794–11803 (2021)

[112] Wang, H., Cong, Y., Litany, O., Gao, Y., Guibas, L.J.: 3dioumatch: Leveraging iou prediction for semi-supervised 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14615–14624 (2021)

[113] Wang, T., Pang, J., Lin, D.: Monocular 3d object detection with depth from motion. In: European Conference on Computer Vision (ECCV) (2022)

[114] Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: CVPR (2019)

[115] Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) (2019)

[116] Wang, Z., Jia, K.: Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1742–1749. IEEE (2019)

[117] Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big Data **3**(1), 1–40 (2016)

[118] Xia, Q., Yang, J.J.: Memristive crossbar arrays for brain-inspired computing. Nature Materials **18**, 309–323 (2019)

[119] Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves ImageNet classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10687–10698 (2020)

[120] Xu, C., Li, T., Tang, C., Sun, L., Keutzer, K., Tomizuka, M., Fathi, A., Zhan, W.: Pretram: Self-supervised pre-training via connecting trajectory and map. arXiv preprint arXiv:2204.10435 (2022)

[121] Xu, Y., Zhu, X., Shi, J., Zhang, G., Bao, H., Li, H.: Depth completion from sparse lidar data with depth-normal constraints. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)

[122] Xuu, S., Zhou, D., Fang, J., Yin, J., Bin, Z., Zhang, L.: Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. In: ITSC (2021)

[123] Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10) (2018). https://doi.org/10.3390/s18103337, `https://www.mdpi.com/1424-8220/18/10/3337`

[124] Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 7652–7660 (2018), `https://api.semanticscholar.org/CorpusID:52238978`

[125] Yang, Z., Zhou, Y., Chen, Z., Ngiam, J.: 3d-man: 3d multi-frame attention network for object detection. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1863–1872 (2021)

[126] Yao, X.S., Maleki, L.: Optoelectronic oscillator for photonic systems. IEEE Journal of Quantum Electronics **32**(7), 1141–1149 (1996)

[127] Yin, T., Zhou, X., Krähenbühl, P.: Center-based 3d object detection and tracking. CVPR (2021)

[128] Yin, T., Zhou, X., Krähenbühl, P.: Multimodal virtual point 3d detection. NeurIPS (2021)

[129] You, Y., Diaz-Ruiz, C.A., Wang, Y., Chao, W.L., Hariharan, B., Campbell, M., Weinberger, K.Q.: Exploiting playbacks in unsupervised domain adaptation for 3d object detection in self-driving cars. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 5070–5077 (2022)

[130] Yuan, Y., Kitani, K.: Dlow: Diversifying latent flows for diverse human motion prediction. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)

[131] Yuan, Y., Weng, X., Ou, Y., Kitani, K.: Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)

[132] Zhang, A., Brown, L.D., Cai, T.T.: Semi-supervised inference: General theory and estimation of means. Annals of Statistics **47**, 2538–2566 (2019)

[133] Zhang, X., Kwon, K., Henriksson, J., Luo, J., Wu, M.C.: A large-scale microelectromechanical-systems-based silicon photonics lidar. Nature **603**, 253–258 (2022)

[134] Zhao, N., Chua, T.S., Lee, G.H.: Sess: Self-ensembling semi-supervised 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)

[135] Zhao, Q., Zhu, B.: Towards the fundamental limits of knowledge transfer over finite domains. arXiv preprint arXiv:2310.07838 (2023)

[136] Zhou, H., Ge, Z., Liu, S., Mao, W., Li, Z., Yu, H., Sun, J.: Dense teacher: Dense pseudo-labels for semi-supervised object detection. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 35–50 (2022)

[137] Zhou, Q., Yu, C., Wang, Z., Qian, Q., Li, H.: Instant-teaching: An end-to-end semi-supervised object detection framework. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4079–4088 (2021)

[138] Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. In: arXiv preprint arXiv:1904.07850 (2019)

[139] Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: CVPR (2018)

[140] Zhu, B., Ding, M., Jacobson, P., Wu, M., Zhan, W., Jordan, M., Jiao, J.: Doubly-robust self-training. In: Advances in Neural Information Processing Systems. vol. 36, pp. 41413–41431 (2023)

[141] Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3d object detection. arXiv preprint arXiv:1908.09492 (2019)

[142] Zhu, X.J.: Semi-supervised learning literature survey (2005)