ABSTRACT

Title of Dissertation: LEARNING-BASED AUTONOMOUS DRIVING WITH ENHANCED DATA EFFICIENCY AND POLICY TRAINING

Yu Shen
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Ming C. Lin
Department of Computer Science

Autonomous vehicles are capable of sensing the environment and moving around with little to no human intervention, enhancing efficiency and safety. Self-driving cars, for instance, will affect our modes of transportation and way of life in the years to come. With rapid advances in hardware and software design, learning-based autonomous driving is becoming a viable and popular solution.

This thesis focuses on data from the front-end and policy from the back-end in learning-based autonomous driving. As commonly known, data is central to all learning-based methods. However, engineers cannot collect all the data from all possible scenarios to train a model due to the great variety of real-world driving scenarios, e.g., different conditions in weather, lighting, roads, traffic, etc. Consequently, the issue of how to train a model with robustness, generalizability, and transferability becomes crucial. In addition, while input data is the key component of autonomous driving in the front-end, policy, which controls the vehicle to navigate safely, is also an essential component in the back-end. Existing methods have made

progress on policy learning, but there is room for improvement, e.g., reinforcement learning is not able to utilize expert demos, inverse reinforcement learning can not directly utilize driving domain knowledge, etc. I propose to address these important open research issues by adopting machine learning and deep learning techniques, including adversarial data augmentation and training, auxiliary modality learning, transfer learning, reinforcement learning, and inverse reinforcement learning.

To make autonomous driving more robust against varying weather changes, lighting conditions, or other image corruptions, I propose a gradient-free adversarial training method based on data augmentation and sensitivity analysis. To utilize multi-modal information for good performance with low computational costs, I design an auxiliary modality learning framework that can distill knowledge from multi-modality data to single modality data, with a specific condition that allows the teacher network to stay aware of the student's status for better distillation. I further propose a small-shot cross-modal distillation to solve the problem in a small-shot setting. To overcome the difficulty of collecting data in the real world, I present a transfer learning architecture that is able to transfer knowledge from the simulation domain to the real-world scenarios. To utilize both expert demonstration and real-world driving knowledge, I propose enhanced inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning.

In summary, my proposed learning-based frameworks enhance robustness, efficiency, and generalization via adversarial data augmentation and training, auxiliary modality learning, and transfer learning w.r.t. data processing in the front-end; and further improve policy learning via inverse reinforcement learning in the back-end.

LEARNING-BASED AUTONOMOUS DRIVING WITH ENHANCED DATA EFFICIENCY AND POLICY TRAINING

by

Yu Shen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:
Professor Ming C. Lin, Chair
Professor Mumu Xu, Dean's Representative
Professor Tom Goldstein
Professor Furong Huang
Professor Dinesh Manocha
Professor Tianyi Zhou

Acknowledgments

I would like to thank my advisor, Prof. Ming C. Lin. I was led to the fascinating world of autonomous driving and managed to contribute to the cutting-edge technology thanks to her guidance. I would also want to thank my committee members who provide a lot of useful comments and insights to me to improve the dissertation.

It is not possible to finish this journey without all the kind help and support from my all of my friends and all the GAMMA members. Last but not the least, I would like to thank my parents, my sister and her husband, and my girlfriend for encouraging me to pursue my dream and giving me support all the time.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation and Overview

Autonomous driving, as a vital component of the transportation system, has the potential to become one of the life-altering revolutionary technologies. With little to no human input, autonomous vehicles is able to sense, predict, plan, and move safely, which enhances the safety and efficiency of the transportation system. Autonomous driving is increasingly adopted in real-world applications, e.g. autonomous truck for cargo transportation, self-driving taxi in urban areas, etc. With rapid advancements in high-performance computing hardware (e.g., GPU, TPU) and deep learning foundation (e.g., theoretical proof, network architecture design, and training paradigm design) and software (e.g., deep learning languages, platforms, libraries), learning-based autonomous driving is becoming a main-stream solution in autonomous vehicles. As commonly known, data is central to all learning-based methods. We aim to improve performance by utilizing self-augmented data (*data augmentation and adversarial learning* in Chapter 2), other modalities' data (*multi modality learning and auxiliary modality learning* in Chapter 3, 4), and other domains' data (*transfer learning and domain adaptation*, Chapter 5). In addition, while input data is the key component of autonomous driving in the front-end, pol-

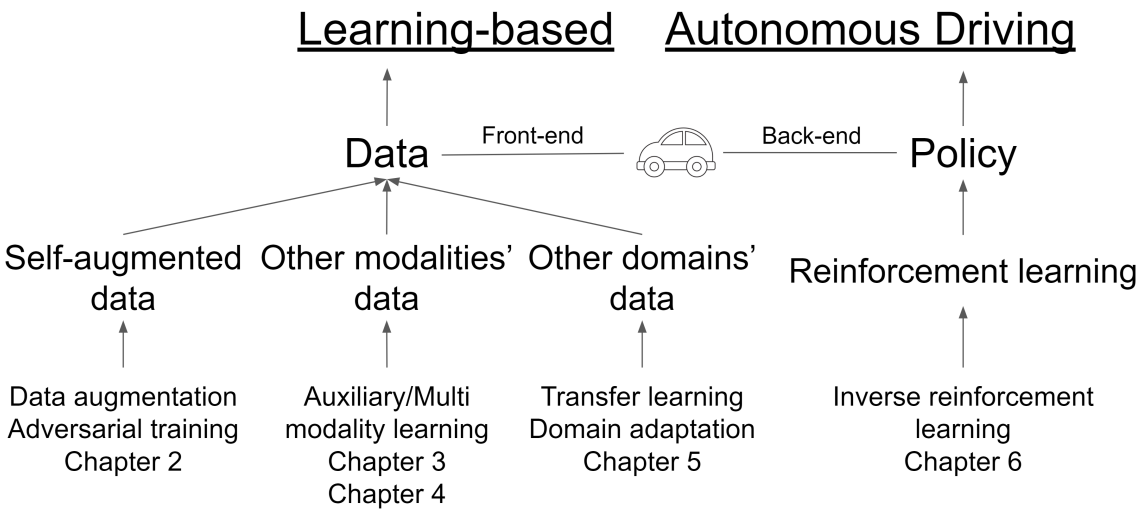


Figure 1.1: Dissertation Structure Overview.

icy, which controls the vehicle to navigate safely, is also an essential component in the back-end. We thus address the issue on policy learning with *enhanced inverse reinforcement learning* (Chapter 6). An overview of my dissertation can be found in Fig. 1.1.

Data, as the knowledge source, plays an vital role in all learning-based methods, including autonomous driving. In autonomous driving, various types of data come from different sensors, e.g., RGB image from RGB camera, point cloud from lidar, distance reading from radar, event image from event camera, rotation and acceleration from IMU, etc. Different data has different properties, and is suited to various scenarios. For example, the RGB camera is the most popular sensor due to its low cost and rich visual information that can be used to solve common perception tasks, but it may fail in the dark. Lidar can detect distance accurately and output a 3D point cloud, but it is expensive and cannot detect texture information. Therefore, I explore how to maximize the benefits of multi-modality, while

minimizing potential drawbacks, for autonomous driving. Aside from direct sensing data, there are techniques for generating new data, e.g., data augmentation, simulator, etc. Typically, Data augmentation generates new data by applying changes to the original data, e.g., cropping, rotating, translating, and flipping, are standard techniques to augment image data for classic computer vision tasks like image classification. Data augmentation can also be used to improve model robustness or accuracy. Simulators typically generate synthetic data with known physical models, which help generate more data otherwise difficult to collect in the real world like traffic accidents. However, there exists a domain gap between virtual and real world. Thus, how to transfer knowledge from the simulation domain to the real world, or from one domain to another, has become a prevalent challenge.

Policy, as the control command generator, also plays a crucial role in autonomous driving. By providing sensing data, policy is able to generate the command for the autonomous vehicle to execute. The two most prevalent techniques for learning policy are reinforcement learning and inverse reinforcement learning. Reinforcement learning can take advantage of task domain knowledge by designing a reasonable reward function. However, designing a perfect reward function can be challenging when we only have demonstrations from other experts. Inverse reinforcement learning, on the other hand, can use expert demonstration in place of designing a reward function. However, it is difficult to take advantage of domain knowledge or prior information into account. Additionally, it is time-consuming, because multiple rounds of reinforcement learning are involved. Finding solutions to overcome those issues can further propel advances in autonomous driving.

In this dissertation, I propose to address the key challenges by adopting machine learning for autonomous driving, specifically deep learning techniques, i.e. adversarial data augmentation and training, auxiliary modality learning, transfer learning, reinforcement learning and inverse reinforcement learning. These include:

- Robust autonomous driving by sensitivity analysis and adversarial training (Chapter 2);
- Low cost autonomous driving by distilling knowledge from multi-modality data to single modality data (Chapter 3, 4));
- Generalized autonomous driving, transfer learning from virtual to real, or from one domain to another (Chapter 5);
- Better policy learning in autonomous driving via inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning (Chapter 6).

1.2 Data for Learning-based Autonomous Driving

Data is critical for all learning-based methods, as well as autonomous driving. In this section, we mainly discuss self-augmented data for vision robustness, other modalities' data for auxiliary modality learning, and other domains' data for transfer learning.

1.2.1 Self-augmented Data for Vision Robustness in Autonomous Driving

Autonomous driving is a complex task that requires many software and hardware components to operate reliably under highly disparate and often unpredictable conditions. While on the road, vehicles will encounter day and night, clear and foggy conditions, sunny and rainy days, as well as bright cityscapes and dark tunnels. All these external factors in conjunction with internal factors of the camera (e.g., those associated with hardware like camera distortion) can lead to quality variations in input data for image-based learning algorithms. These image corruptions or perturbations may lead to performance drop of the model.

To improve the robustness of learning-based methods, techniques like data augmentation [1] and adversarial learning [2, 3, 4] have become popular. Typically, data augmentation generates new data by applying changes with a specific pattern to the original data, e.g., cropping, rotating, translating, and flipping, are standard techniques for augmenting image data for classic computer vision tasks like image classification. It can fortify machine learning systems against these degradations by simulating them before or during training. Adversarial learning usually applies changes to original data during training and selects the worst variant for backpropgation, resulting in the most remarkable improvement.

Researchers have investigated how to improve the robustness of learning algorithms in the presence of varying image quality degradations [5, 6, 7, 8, 9, 10, 11, 12, 13]. However, an algorithmic tool for analyzing the sensitivity of real-world neural

network performance on the properties of (and corruptions to) training images is lacking. More importantly, a mechanism to leverage such a sensitivity analysis for improving learning outcomes needs to be developed.

1.2.2 Other Modalities' Data for Auxiliary Modality Learning in Autonomous Driving

Due to many different scenarios arising from driving, auxiliary information is often considered in addition to single sensor information to improve learning of the autonomous vehicles. Many existing autonomous driving solutions depend on multi-modality solutions, where input data originates from variety of sensors, e.g., RGB camera, Lidar, radar, etc. RGB cameras have become the main sensor choice because of its abundant availability and low cost. While other sensors like Lidar do provide useful information, e.g. depth that can help better understand the environment, their cost is typically quite high. Previous works [14, 15, 16] have attempted to exploit depth information in addition to the RGB channels. The unified learning framework that involves multiple modalities of data as input is referred as multi-modality learning. However, it is computationally expensive. Also, the framework that requires the auxiliary sensor/data for input at test time has been limited in its application. For instance, it is expensive to deploy Lidar on commodity self-driving cars, but it's reasonable to equip a few developer's cars with Lidar for solely training purposes.

Since using all modalities during inference in specific applications is difficult,

some works consider employing multiple modalities during training but fewer modalities during evaluation and inferencing. However, this form of learning tasks, i.e., "test with fewer modalities than during training", is not standardized yet. For example, there is no formal term or definition. There have been concepts, such as "learning with side information" [17], "learning with privileged information" [18], "learning with auxiliary modality" [19], "learning with partial-modalities" [20], and "modality distillation" [21], etc. We formalize these learning tasks as *Auxiliary Modality Learning (AML)*. Furthermore, AML can be achieved in different variants. For example, the amount of auxiliary information may be limited in some cases. Or, it may require expensive expert-labeled data, or need sensing data from a low-frequency sensor that makes the network harder to learn. A systematic analysis is needed AML.

1.2.3 Other Domains' Data for Transfer Learning in Autonomous Driving

Autonomous driving (AD) has the potential to create safer and more efficient transportation systems by reducing congestion and accidents, due to human errors. Central to AD is a complex task of autonomous steering that requires the choreography of many components to operate. One essential component is the perception-control module that maps sensor data to control commands (e.g., setting steering angles). With recent advances in machine learning, especially deep learning [22], the perception-control module is increasingly enabled by learning-based algorithms,

which leverage multimodal input from sensors including cameras, Lidar, and radar to navigate autonomous vehicles (AVs). Despite each type of sensor offering its unique strength in detecting the environment, the camera is one of the most universal and accessible sensors due to its rich visual information and affordable cost.

As a result, many real-world images are collected for training AVs. Example datasets include KITTI [23], NVIDIA [24], Waymo Open Dataset [25], CityScapes [26], and BDD100K [27]. Apart from real-world images, simulators and virtual images are also heavily used in training AVs [28]. Example simulation platforms include CARLA [29], the Udacity Self-Driving Car Simulator [30], and NVIDIA Drive Constellation [31]. Many scenarios that are crucial for testing autonomous driving, but difficult to capture in the real world, can be modeled in the virtual world at ease, e.g., accidents. While virtual images are believed to supplement real images, the domain gap between the two can obstruct the conjecture. Works have attempted to bridge this domain gap [32, 33, 34, 35, 36, 37]. Furthermore, the recent advancement of image style transfer techniques [38] such as CycleGAN [39] and MUNIT [40] challenges the domain gap and has raised new conjectures on whether realistic-looking images converted from virtual images can be used for learning [41]. Finally, the data from one real scenario also offer benefits to train a model in another real scenario, because of the common information shared between the two scenarios, e.g., drive along the road.

1.3 Policy for Learning-based Autonomous Driving

Policy, as the control command generator, plays a crucial role in autonomous driving. A good policy is able to navigate the autonomous vehicle to the goal efficiently and safely without collision. Reinforcement learning and inverse reinforcement learning are two of the most common techniques to learn policy.

1.3.1 Inverse Reinforcement Learning in Autonomous Driving

Reinforcement learning can take advantage of task domain knowledge by designing reasonable reward functions, but it can be challenging to design a perfect reward function when we only have demonstrations from other experts. It enables the autonomous vehicle to learn policy by exploring the environment (typically in a simulator) and receiving different rewards in different states, and encourages the model to maximize the reward so that the policy will behave as desired. In contrast, inverse reinforcement learning is able to use expert demonstrations instead of designing a reward function. As an effective technique for imitation learning, IRL involves two steps: 1) learning a reward function from expert demonstrations and 2) using the acquired reward function for RL to learn a control policy [42]. To provide some examples, Sharifzadeh et al. [43] apply Deep Q-Networks to extract a reward function in a large state space. You et al. [44] use deep neural networks to approximate the latent reward function of the expert and then apply deep Q-learning to obtain the control policy. However, it is challenging to take advantage of domain knowledges or prior information into account. Also, it is time-consuming because

multiple rounds of reinforcement learning are involved. Recently, curriculum learning for RL has also gained much attention [45, 46]. Some well-known works such as AlphaGo [47] use curricula implicitly to guide training.

1.4 Outcome

In this section, thesis statement is firstly proposed, followed by main results to support each component of the thesis statement.

1.4.1 Thesis Statement

Learning-based autonomous driving can be enhanced in robustness, efficiency, and generalization through adversarial data augmentation and training, auxiliary modality learning, and transfer learning, w.r.t. data processing in the front-end and improved policy learning using inverse reinforcement learning in the back-end.

To support this thesis statement, I present five novel algorithms:

1. A gradient-free adversarial training scheme for enhancing the robustness of autonomous driving against various combinations of image degradations (Chapter 2);

2. An auxiliary modality learning algorithm to achieve cross modality distillation (Chapter 3);

3. A framework that distills modality knowledge in a partially available auxiliary modality setting (Chapter 4);

4. A framework to solve domain-agnostic learning with domain augmentation, feature decomposition, and curriculum learning (Chapter 5);

5. An efficient IRL framework consisting of hybrid-weight trust-region optimization and curriculum learning (Chapter 6).

1.4.2 Main Results

This dissertation presents five methods to support each component of the thesis statement. I list the main results obtained within these methods below:

1.4.2.1 Gradient-Free Adversarial Training Against Image Corruption for Learning-based Steering

In Chapter 2, I introduce a simple yet effective framework for improving the robustness of learning algorithms against image corruptions for autonomous driving. These corruptions can occur due to both internal (e.g., sensor noises and hardware abnormalities) and external factors (e.g., lighting, weather, visibility, and other environmental effects). Using sensitivity analysis with FID-based parameterization, I propose a novel algorithm exploiting basis perturbations to improve the overall performance of autonomous steering and other image processing tasks.

The key contributions of this work include:

- A gradient-free adversarial training scheme for enhancing the robustness of autonomous driving against various combinations of image degradations;

- A systematic method for measuring the severity of image degradation and predicting the impact of such degradation on model performance with Fréchet Inception Distance (FID).

1.4.2.2 Auxiliary Modality Learning with Generalized Curriculum Distillation

In Chapter 3, I propose a new AML method using generalized curriculum distillation to enable more effective curriculum learning. This method includes a new supermodel condition that allows the teacher network to be aware of the student's status in a curriculum way, leading to a better distillation.

The key contributions of this work include:

- Systematically list and classify different types of auxiliary modalities and architectures for AML, and analyze the performance behavior of different types of auxiliary modalities and architectures for AML across different datasets, backbones and tasks;

- Propose a novel AML method, "Smart Auxiliary Modality Distillation (SAMD)", that automatically (1) chooses the best auxiliary modality for the main distillation process, and (2) performs knowledge distillation under a special "supermodel condition" to enable the teacher network to be aware of the student's status;

- Analyze and explain the reasons for the effectiveness of AML from both optimization perspective and data perspective, providing theoretical support to

the SAMD method.

1.4.2.3 Small-shot Multi-modal Distillation for Vision-based Autonomous Steering

In Chapter 4, I propose a novel learning framework for autonomous systems that uses a small amount of "auxiliary information" that complements the learning of the main modality, called "small-shot auxiliary modality distillation network (AMD-S-Net)". The AMD-S-Net contains a two-stream framework design that can fully extract information from different types of data (i.e., paired/unpaired multi-modality data) to distill knowledge more effectively.

The key contributions include:

- a novel framework that distill knowledge from multi-modality model to single-modality model in a partially available auxiliary modality setting, which contains a specific framework design to fully distill the information, i.e., *consistency supervision* for the pairwise data and *distribution divergence supervision* for unpaired data;

- a novel knowledge distillation training paradigm that enables teachers to explore and learn student's local loss landscape information in a higher dimension, thus making it feasible to help student get out of local minimal and boost performance, based on a special "reset operation" that allows the teacher to be aware of the exact student states.

1.4.2.4 Task-Driven Domain-Agnostic Learning for Autonomous Steering

In Chapter 5, I propose a novel framework to solve domain-agnostic learning with domain augmentation, feature decomposition, and curriculum learning. Specifically, we use (1) domain-specific adapters and shared modules to disentangle *domain-specific* information and *task-specific* information; (2) style-transferred branch to help extract domain-specific information; (3) gradually increased ratio of target domain data in each epoch for better knowledge transfer from source to target domain.

The key contributions of this work include:

- A novel framework to solve domain-agnostic learning in the end-to-end steering task, with specific design with domain augmentation, feature decomposition, and curriculum learning;
- Analysis of how different factors influence the end-to-end steering task, including training data (image style, data amount from source and target domain), network architecture (Batch Norm layers, Adapters), and training paradigm (finetuning, partially finetuning, reinitialization).

1.4.2.5 Inverse Reinforcement Learning with Hybrid-weight Trust-region Optimization and Curriculum Learning for Autonomous Maneuvering

In Chapter 6, I propose a novel framework of inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning for autonomous maneuvering. This method can incorporate both expert demonstration (from real driving) and domain knowledge (hard constraints such as collision avoidance, goal reaching, etc. encoded in reward functions) to learn an effective control policy. The hybrid-weight trust-region optimization is used to determine the difficulty of the task curriculum for fast incremental curriculum learning and improve the efficiency of inverse reinforcement learning by hybrid weight tuning of different sets of hyperparameters.

The key contributions of this work include:

- Hybrid-weight trust-region optimization improves upon IRL [42] by imposing non-uniform priors on task-critical features, e.g., collision avoidance and goal-seeking, incorporating *both* expert demonstration and domain knowledge to automate weight tuning effectively;

- Curriculum learning retains important lessons through *increasingly difficult* RL to task learning, thus improving overall training efficiency and performance;

- Curriculum learning also utilizes the hybrid-weight trust-region optimization

to *assess curriculum difficulty*;

- Our framework is further compatible with domain-dependent techniques, such as learn-from-accident [48], which generate safe trajectories, further boosting the overall performance in autonomous driving.

1.5 Outline of Dissertation

The subsequent chapters of this dissertation are organized as follows.

Chapter 2 introduces a simple yet effective framework for improving the robustness of learning algorithms against image corruptions for autonomous driving.

Chapter 3 presents a new AML method using generalized curriculum distillation to enable more effective curriculum learning.

Chapter 4 demonstrates a novel learning framework for autonomous systems that uses a small amount of "auxiliary information" that complements the learning of the main modality, called "small-shot auxiliary modality distillation network (AMD-S-Net)".

Chapter 5 presents a novel framework to solve domain-agnostic learning with domain augmentation, feature decomposition, and curriculum learning.

Chapter 6 describes a novel framework of inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning for autonomous maneuvering.

Chapter 2: Gradient-Free Adversarial Training Against Image Corruption for Learning-based Steering

2.1 Introduction

Autonomous driving is a complex task that requires many software and hardware components to operate reliably under highly disparate and often unpredictable conditions. In this work, we study "learning-based steering" as it contains both perception and control, both critical components for autonomous driving. While on the road, vehicles are going to experience day and night, clear and foggy conditions, sunny and rainy days, as well as bright cityscapes and dark tunnels. All these external factors in conjunction with internal factors of the camera (e.g., those associated with hardware) can lead to quality variations in input data for image-based learning algorithms. One can harden machine learning systems to these degradations by simulating them at training time [28]. However, an algorithmic tool for analyzing the sensitivity of real-world neural network performance on the properties of (and corruptions to) training images is lacking. More importantly, a mechanism to leverage such a sensitivity analysis for improving learning outcomes needs to be developed. In this work, we quantify the influence of image quality on the task of "learning

to steer," study how training on degraded and low-quality images can boost robustness to image corruptions, and provide a systematic approach to improve the performance of learning algorithms using quantitative analysis.

Image degradations can be simulated by varying attributes such as blur, noise, distortion, color representations (such as RGB or CMY) hues, saturation, and intensity values (HSV). However, identifying the correct combination of the simulated corruptions to obtain optimal performance on real data is a difficult—if not impossible—task, as it requires domain transfer and exploring a high dimensional parameterized space.

We *design a systematic method for measuring the severity of image degradation and predicting the impact of such degradation on model performance.* Inspired by the use of image feature variance in sensitivity analysis [49], we measure the difference between real-world image distributions and simulated/degraded image distributions using Fréchet Inception Distance (FID). Our results confirm that FID can help predict the performance of a model trained using simulated data and deployed in the real world. Next, we use FID between different simulated datasets as a unified metric to parameterize the severity of various image degradations due to different factors.

Borrowing concepts from the adversarial attack literature [2, 3, 4], we *build a scalable training scheme for enhancing the robustness of autonomous driving against various combinations of image degradations, while* increasing *the overall accuracy of the steering task on clean data.* Our proposed method constructs a dataset of adversarially degraded images by applying optimization within the space of possible

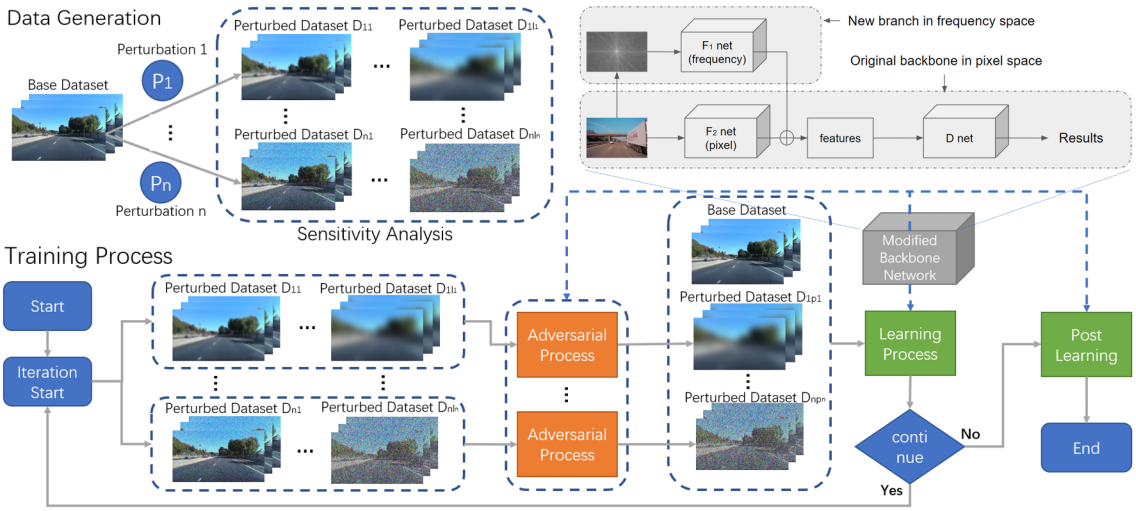


Figure 2.1: Pipeline of our method. **Data generation**. We generate perturbed datasets of each factor at multiple levels based on the FID-parameterized sensitivity analysis results. **Training process**. First stage: in each iteration, we first augment the training dataset with "adversarial images" generated by applying an image corruption; we then combine the base and perturbed datasets to train our model to maximize the overall performance. A frequency-space branch is added to the backbone when frequency-related perturbations (e.g., blur, noise) need to be handled. Second stage: in post-learning, the model is fine-tuned solely on clean data to boost accuracy, while performing validation on both clean and perturbed data with an early break when overall performance decreases to maintain the performance on the perturbed data.

degradations during training. As shown in Fig. 2.1, the method begins by training on a set of real and simulated/degraded images using arbitrary degradation parameters. During each training iteration, the parameters are updated to generate a new degradation set so that the model performance is (approximately) minimized. The network is then trained on these adversarially degraded images to promote robustness. A post-training step is applied to further improve the performance on clean data without weakening robustness. Our proposed algorithm uses our FID-based parameterization to discretize the search space of degradation parameters and accelerates the process of finding optimal parameters.

Experiments show that our algorithm improves the performance of "learning

to steer" up to **97%** in mean accuracy over baselines, and especially improves the performance on datasets contaminated with complex combinations of perturbations (up to **87%**). It additionally boosts the test performance on degradations that are not seen during training, including simulated snow, fog, and frost (up to **77%**). We also compare our approach with other SOTA techniques (e.g., data augmentation and adversarial training) on visual processing tasks such as detection and classification. Our method consistently achieves higher performance. In addition, our method is easy to implement and can be readily integrated with other frameworks such as object detection, classification, regression, etc.

Finally, we propose a comprehensive robustness evaluation standard under four different scenarios: clean data, single-perturbation data, multi-perturbation data, and previously unseen data. While state-of-the-art studies usually conduct testing under one or two scenarios (e.g., ImageNet-C [50]), our work tests and verifies results under four meaningful scenarios. We plan to release code and datasets for benchmarking "autonomous driving under perturbations" using unseen factors such as image corruptions in ImageNet-C [50], totaling **480** datasets and **26M** images.

2.2 Related Work

The influence of noise and distortion on real images for learning tasks has been well explored. For example, researchers have examined the impact of optical blur on convolutional neural networks and present a fine-tuning method for recovering lost accuracy using blurred images [51]. This fine-tuning method resolves lost accuracy

when images are distorted instead of blurred [52]. While these fine-tuning methods are promising, Dodge and Karam [53] find that tuning to one type of image quality reduction effect would cause poor generalization to other types of effects. The comparison of image classification between deep neural networks and humans shows similar performance on good-quality images [54]. However, deep neural networks struggle significantly more than humans on low-quality, distorted, and noisy images. One study shows that adversarial perturbations are more prevalent in the Y channel in the YCbCr color space of images than the other two channels, while perturbations in RGB channels are equally distributed [55]. Wu et al. [56] studies the effect of Instagram filters on learning tasks, and introduces a lightweight de-stylization module that predicts parameters used for scaling and shifting feature maps to "undo" the changes incurred by filters.

Researchers have also explored how to improve the robustness of learning algorithms under various image quality degradations. One recent work provides a novel Bayesian formulation for data augmentation [5]. Cubuk et al. [6] proposes an approach to automatically search for improved data augmentation policies. Ghosh et al. [7] analyzes the performance of convolutional neural networks on quality degradations due to compression loss, noise, blur, and contrast, and introduces a method to improve the learning outcome. Another work [8] shows that self-supervision techniques can be used to improve model robustness and exceeds the performance of fully supervised methods. A recent method, AugMix [9] improves model robustness using data augmentation, where transformation compositions are used to create a new dataset that is visually and semantically similar to the original dataset. AugMix

is compared to several other augmentation methods, which comprise Cutout [10], MixUp [11], and CutMix [12]. Gao et al. [13] proposes a technique to re-purpose software testing methods to augment the training data of DNNs, with the objective to improve model robustness. A recent work improves model generalizability by first augmenting the training dataset with random perturbations, and then minimizing worst-case loss over the augmented data [57].

Our work differs from these studies in several regards. First, we simulate adversarial conditions of image factors instead of using commonplace image conditions. Second, we conduct a systematic sensitivity analysis for preparing datasets that are representative of image degradations from multiple factors at various levels. Third, our algorithm can work with the discretized parameter space while generalizing well to the continuous parameter space. Another advantage of our approach is that we can augment the training dataset without the derivatives of the factor parameters, which may not exist or are difficult to compute.

2.3 Background and Setup

Task. Our target task is end-to-end steering: given a single image as input (e.g. captured by a front-facing camera on a self-driving car), output a steering angle that drives the car safely on the road [58]. A steering angle of 0 represents the forward direction. We use mean accuracy (MA) to evaluate the task since it represents overall performance under a variety of measures (See Sec. 2.5).

Datasets. We choose four real-world driving corpuses as our datasets: Audi [59],

Honda [60], Waymo [25], and SullyChen [24]. The Audi dataset is the most recent (2020); the Honda dataset has 100+ long-time driving videos; Waymo includes many environmental conditions such as weather and lightening and is a large dataset (390k frames for perception); and the SullyChen dataset focuses on the steering task and has the longest continuous driving image sequence without road branching. The details of these datasets are provided in Appendix 2.7.3. Other datasets, such as CIFAR-100, for various computer vision tasks are also used to demonstrate the generalizability of our technique.

Basis perturbation. We study nine basis perturbations: blur, noise, distortion, three-color (RGB) channels, and hues, saturation, and intensity values (HSV). Blur, noise, and distortion are among the most commonly used perturbations that can directly affect image quality. R, G, B, H, S, V channels are chosen because they are frequently used to represent image color spaces: RGB represent three basic color values of an image, while HSV represent three common metrics of an image. Other color spaces such as HSL or YUV have similar properties, hence are excluded.

We use Gaussian blur [61] (parameterized by standard deviation), additive white Gaussian noise (AWGN) (parameterized by standard deviation), and radial distortion [62, 63] (with radial distortion parameters $k1, k2$). For representing channel-level perturbations, we use a linear model: denote the value range of one channel C as $[a_C, b_C]$, in the darker direction, we set $v'_C = \alpha a_C + (1-\alpha)v_C$, while in the lighter direction, we set $v'_C = \alpha b_C + (1-\alpha)v_C$, where α is the severity parameter, v_C is the pixel value on clean image and v'_C is the perturbed pixel value. The default values are $a_C = 0$ and $b_C = 255$, with two exceptions. We set $a_V = 10$ to exclude a

Figure 2.2: Example images of quality degradation. Left: Five levels of quality reduction using the blur, noise, and distortion effects (three levels are shown). Right: The original images are shown in the middle row. The top row shows examples in the lighter direction per channel for R, G, B, H, S, V, while the bottom row shows examples in the darker direction per channel.

complete dark image on V channel, and $b_H = 179$ according to H channel definition. See examples in Fig. 2.2 and detailed description in Appendix 2.7.2.

Test scenarios. While other studies usually test in only one or two scenarios [50], we test the performance of all methods in four scenarios with increasing complexity. Scenario 1: *Base dataset*. Test on the base dataset. Scenario 2: *Single perturbation*. Test on the datasets with perturbations, but each dataset only experiences one level of one perturbation (within blur, noise, distortion, R, G, ,B, H, S, V). Scenario 3: *Combined perturbations*. Test on the datasets with combinations of various perturbations at different levels, and each dataset has different levels of all perturbations. Scenario 4: *Unseen perturbation*. Test on the datasets with unseen perturbations at different levels. Specifically, we use "motion blur", "zoom blur", "pixelate", "jpeg compression", "snow", "frost", "fog" from ImageNet-C [50]. We apply the same perturbations/levels to all images within one dataset. (See details in Appendix 2.7.1).

2.4 Improving Robustness of Learning-based Steering

In this section, we introduce our gradient-free adversarial training method to improve the robustness of learning-based steering using Sensitivity Analysis. Our method is among the first to *treat complex unforeseen perturbations as a functional combination of multiple simple "basis perturbations"*. As a result, the robustness of a model to several individual perturbations can lead to robustness against combined or unseen perturbations. In addition, *via discretized sensitivity analysis*, our approach is the first to *use FID to enable cross-factor performance comparison*, i.e., making task sensitivity w.r.t different perturbations comparable. Meanwhile, sensitivity analysis helps minimize the discretization level number, thus speed up the adversarial training process.

2.4.1 Basis Perturbations

There are various types of image corruptions due to different environmental effects or sensor variations, and often these corruptions can be approximated using a combination of basis perturbations, e.g., snow/frost may be simulated using water drop corruptions, Gaussian blur, and image whitening. Inspired by the concept of *basis function*, we explore the possibility of training a model on a few "basis perturbations" and the resulting model is robust against complex perturbations.

Our basis perturbation contains "basis" representations of the color space in RGB and of image metrics in HSV, which span color and intensity/brightness spaces, respectively. Blur and noise are designed to produce low and high frequency corrup-

tions, while distortion captures the 2D positional transformation that may appear from small vibrations and motions of a camera. See comparison of different basis perturbations in Appendix. 2.7.2. We also introduce two metrics to evaluate how well a training method can enable a model to perform on combined perturbations and previously unseen perturbations when learning only on single perturbations. Using basis perturbations to represent image corruption, we then design an algorithm to train neural networks to cope with quality-degraded images as input for autonomous driving.

2.4.2 Sensitivity Analysis

We use an adversarial training process in which we optimize corruption parameters to maximally disrupt model performance. If this was done using gradient optimization, it would require us to differentiate through the corruption operator to update the parameters describing the corruption. Not all gradients of basis perturbations can be easily derived, and corruptions such as distortions may have parameters with non-differentiable effects. Rather than rely on gradient optimization, we introduce an simple, discretized gradient-free method. We use Sensitivity Analysis (SA) to evenly discretize the space of parameters; we choose a discrete subset of values for each parameter that are equally "far apart" in terms of their impacts on the classifier. This is achieved using the Fréchet Inception Distance (FID) [64] to perform SA across a range of corruption types.

SA is commonly used to understand how the uncertainty of the model output (numeric or otherwise) can be apportioned to different sources of uncertainty of the model input [49, 65]. Here, we use SA to study how distortions to model input (blur, noise, distortion, and RGB/HSV brightness shifts) can be apportioned to model output. We also use SA to quantify the level of degradation caused by an corruption, and prepare datasets that are representative of image qualities at various degradation levels.

We adopt the Fréchet Inception Distance (FID) [64] as a unified metric for our SA (The reasons for adopting FID and a quantitative comparison between FID and other metrics such as L2 norm can be found in Appendix 2.7.4). In addition, we focus on the changes in model performance according to the changes in input and define the sensitivity as the first-order derivative of MA w.r.t FID:

$$sensitivity = \frac{\partial MA^*(R \oplus F(\mathbf{p}))}{\partial FID(R, R \oplus F(\mathbf{p}))},$$

where $MA^*(D)$ is the MA test result on dataset D with the model trained on the base dataset R; $FID(A, B)$ is the FID between dataset A and dataset B (where FID is calculated in its standard formulation using a pretrained InceptionV3 network [64]); $F(\mathbf{p})$ is the perturbation with parameter $\mathbf{p}$ (e.g., the standard deviation of the Gaussian kernel), and $D \oplus F$ denotes the dataset obtained by applying perturbation F to D.

Starting from empirically-selected parameters of each factor, we generate perturbed datasets and compute their corresponding MA using the trained model on

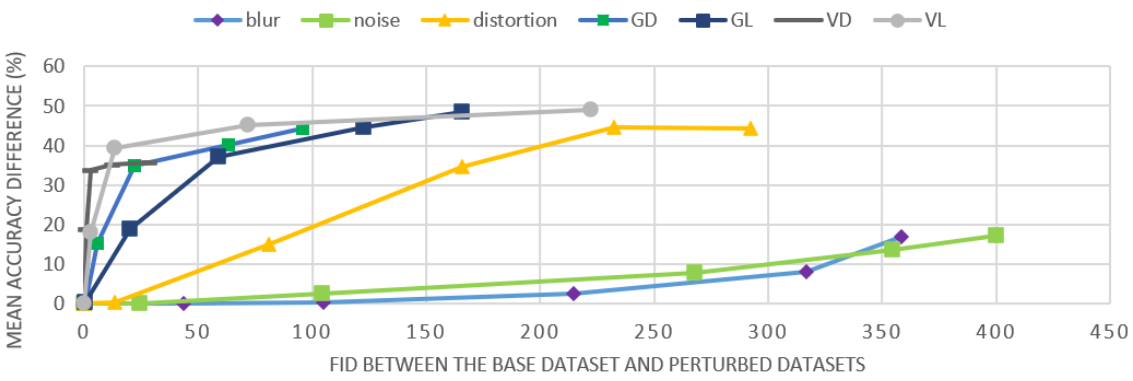


Figure 2.3: The relation between FID and MA differences. GD/GL denotes G channel in darker/lighter direction, and VD/VL denotes V channel in darker/lighter direction, respectively. Sensitivity is represented by the first-order derivative of the curve. Note that the values in the near-zero FID range (i.e., < 50) are more commonly found in real-world scenarios.

R (see numeric results in Appendix 2.7.9). We then map the MAs and their corresponding parameter values into the FID space. By leveraging this new MA-FID space, we aim to minimize the number of sampled parameters for each perturbation for efficient training (see Sec 2.4.3), while preserving similar parameter curves. At a high level, we sample points more densely in the value range that has high sensitivity (closer FID values between sample points), while sampling points sparsely in the value range that has low sensitivity (See specific equation in Appendix. 2.7.4). Examples of the resulting images are shown in Fig. 2.2 (more in Appendix 2.7.1). Detailed descriptions of the final perturbed datasets are provided in Appendix 2.7.2.

The final MA differences in the FID space are visualized in Fig. 2.3 (for blur, noise, distortion, and G and V channels; see the entire figure in Appendix 2.7.4). We first observe that learning-based steering is more sensitive to the channel-level perturbations (i.e., R, G, B, H, S, V) than the image-level perturbations (i.e., blur, noise, distortion). Second, the task is least sensitive to blur and noise but most

sensitive to the V channel (the intensity value). Third, for the same color channel, darker and lighter levels appear to have different MAs at the same FID value. Compared to other "learning to steer" studies [66, 67], our method is the first to transfer perturbations from multiple parameter spaces into one space to enable cross-factor comparison.

2.4.3 Gradient-Free Adversarial Training

Our training process consists of two stages. In the first stage, we use iterative min-max training. At each iteration, we first choose one dataset from the datasets (with different quality degradations) of each basis perturbation (i.e., R, G, B, H, S, V, blur, noise, and distortion) to minimize validation MA, then we merge the nine chosen datasets with the base dataset to train our model while maximizing MA. The first stage stops when a pre-specified number of iterations is reached or the MA loss is below a certain threshold. Our method resembles conventional adversarial training: we improve model robustness by training the model to maximize accuracy using the base dataset plus the perturbed datasets with the minimum accuracy. The loss function is the following:

$$\underset{\theta}{\text{minimize}}\, \underset{\mathbf{p}}{\text{minimize}}\, MA(\theta, U_{\mathbf{p}}),$$

where $\mathbf{p}$ represents the union of all parameter levels of all basis perturbations; θ denotes the model parameters; and $U_{\mathbf{p}}$ is the training dataset. Furthermore, we add a branch in the frequency space to the backbone network to address frequency-

Algorithm 1: Improve robustness of learning-based steering

Result: a trained model parameterized by θ

Pre-processing:

Conduct sensitivity analysis and discretize the parameters of n factors into their corresponding $l_{i=1,...,n}$ levels (Sec. 2.4.2)

Generate new datasets for each factor with the discretized values from the base dataset R (Sec. 2.3 Basis Perturbation): $\{D_{i,j}\}_{i=1,2,..,n,j=1,2,..l_i}$

Initialization:

Initialize $t = 0$, the maximum number of iterations T, the number of epochs k_1 and k_2, and model parameters $\theta^{(0)}$

First stage:

while $t \leq T$ **do**

- For each factor, select D_{i,p_i} that can minimize the validation MA, where $p_i = \arg\min_j MA(\theta^{(t)}, D_{i,j})$
- Merge all selected datasets $U_{\mathbf{p}} = (\bigcup_{i=1}^{n} D_{i,p_i}) \bigcup R$
- Train the network for k_1 epochs and update $\theta^{(t+1)} = \text{train}(\theta^{(t)}, U_{\mathbf{p}}, k_1)$ to maximize $MA(\theta^{(t+1)}, U_{\mathbf{p}})$
- Break if stop conditions are met; otherwise set $t = t + 1$

Second stage:

Train the network with $\theta^{(final)} = \text{train}(\theta^{(T)}, R, k_2)$ and validate the network on $U_{\mathbf{p}}$ (for k_2 epochs or early break if conditions are met)

=0

related perturbations such as blur and noise (See details in Appendix. 2.7.5). The effectiveness is demonstrated in our ablation study (see Table 2.5).

The second stage is to boost "clean" accuracy, where the model is fine-tuned solely on the base dataset. To maintain the performance on perturbed datasets, we terminate this stage if the overall validation accuracy (on both clean and perturbed datasets) start to decrease; otherwise, we continue this stage until it reaches the maximum number of iterations or the MA loss decreases to our expected threshold. See Algorithm 2.4.3. (Detailed explanations and the ablation study of the second stage are provided in Appendix 2.7.6.)

Our method offers several advantages: 1) the training data is augmented with-

out re-train the model, thus improving efficiency; 2) it provides the flexibility to generate datasets at various discretized levels of the factor parameters; 3) it does not require the derivatives of factor parameters; other methods that optimize factor parameters in the continuous space require computing derivatives, which can be difficult (e.g., for distortion); 4) it generalizes to not only unseen parameters of individual factors but also the composition of unseen parameters of multiple factors; and 5) it is easy to implement and can be readily integrated with other frameworks such as object detection, classification, and regression.

2.5 Experiments and Results

Setups. All experiments are conducted using one Intel(R) Xeon(TM) W-2123 CPU, two Nvidia GTX 1080 GPUs, and 32G RAM. We use the Adam optimizer [68] with learning rate 0.0001 and batch size 128 for training. The maximum number of epochs is 2,000. The dataset setup is explained in Sec. 2.3. We use the maximum MA improvement (MMAI), the average MA improvement (AMAI), and mean Corruption Errors (mCE) [50] as the evaluation metrics.

Backbone. We choose the model described in [58] as the main backbone. We select this model as it has been used to steer an autonomous vehicle successfully in both real world [58] and virtual world [48]. In addition, six other networks are tested to show the generalizability of our method.

Evaluation metrics. We define the accuracy w.r.t a threshold τ as $acc_\tau = count(|v_{predicted} - v_{actual}| < \tau)/n$, where n denotes the number of test cases; $v_{predicted}$

and v_{actual} indicate the predicted and ground-truth value, respectively. We compute mean accuracy (MA) as $\sum_{\tau} acc_{\tau \in \mathcal{T}}/|\mathcal{T}|$, where $\mathcal{T} = \{1.5, 3.0, 7.5, 15, 30, 75\}$ contains empirically selected thresholds of steering angles. Lastly, we use the maximum MA improvement (denoted as MMAI), the average MA improvement (denoted as AMAI), and mean Corruption Errors (mCE) [50] as the evaluation metrics.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
Data Augmentation	-0.44	46.88	19.97	51.34	**36.1**	11.97	75.84	27.5	7.92	81.51
Adversarial Training	-0.65	30.06	10.61	74.42	17.89	6.99	86.82	16.9	8.17	89.91
MaxUp	-7.79	38.30	12.83	66.56	26.94	16.01	72.60	23.43	5.54	81.75
AugMix	-5.23	40.27	15.01	67.49	26.81	15.45	68.38	28.70	8.85	87.79
Ours w/o FS	0.93	48.57	20.74	49.47	33.24	17.74	63.81	29.32	9.06	76.20
Ours	**2.12**	**49.97**	**23.92**	**37.30**	33.15	**22.12**	**54.38**	**33.16**	**13.81**	**61.61**

Table 2.1: Performance of different methods against the baseline [58] on SullyChen dataset with different perturbations. We compare the basic data augmentation method (simply combine all perturbed datasets into training), an adversarial training method [69], MaxUp [57], and AugMix [9]. Overall, our method outperforms all other methods (i.e., highest MA improvements and lowest error in mCEs) in practically all scenarios. Notice ours w/o FS (without frequency space) also outperforms other techniques.

Comparison with different methods. We compare our method with four other methods: an adversarial training method [69], a basic data augmentation method, MaxUp [57], and AugMix [9], to see the performance improvement over the baseline method [58]. For the basic data augmentation method, we simply merge all perturbed datasets for model training.

From Table 2.5, we observe that our method outperforms other methods under all metrics in all scenarios: not only on the clean dataset but also on perturbed datasets. Notably, our algorithm improves the performance of "learning to steer" up to 50% in MMAI, while reducing mCE by 60% over the baseline (Scenario 2). Our method also improves the task performance using the combined datasets (Scenario

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
AugMix+Nvidia	-0.12	40.64	10.94	76.48	25.97	16.79	64.41	**22.23**	5.99	84.95
Ours+Nvidia	**2.48**	**43.51**	**13.51**	**67.78**	**28.13**	**17.98**	**61.12**	16.93	**6.70**	**80.92**
AugMix+Comma.ai	-5.25	55.59	9.56	86.31	31.32	**0.77**	**106.1**	37.91	7.97	89.99
Ours+Comma.ai	**0.36**	**62.07**	**15.68**	**70.84**	**38.01**	0.74	108.32	**42.54**	**12.15**	**77.08**
AugMix+ResNet152	-4.23	20.84	1.45	96.24	12.21	6.71	80.19	15.40	2.87	97.62
Ours+ResNet152	**-0.96**	**24.29**	**5.19**	**79.76**	**16.05**	**8.02**	**75.16**	**16.58**	**5.33**	**85.68**

Table 2.2: Performance improvement of different backbones against the baseline performance using the Honda dataset. Our method outperforms AugMix in most cases. Notice that the methods with ResNet152 do not improve as much as the first two networks (since the ResNet152 baseline already has high performance).

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
AugMix on SullyChen	-5.23	40.27	15.01	67.49	26.81	15.45	68.38	28.70	8.85	87.79
Ours on SullyChen	**1.46**	**49.76**	**22.87**	**40.84**	**33.15**	**22.12**	**54.38**	**33.87**	**13.51**	**62.50**
AugMix on Audi	-8.24	81.89	32.22	55.27	75.49	50.23	41.98	73.06	27.39	77.51
Ours on Audi	**5.98**	**97.57**	**48.50**	**10.27**	**87.56**	**62.38**	**25.80**	**77.22**	**32.71**	**39.14**
AugMix on Honda10k	-0.12	40.64	10.94	76.48	25.97	16.79	64.41	**22.23**	5.99	84.95
Ours on Honda10k	**2.48**	**43.51**	**13.51**	**67.78**	**28.13**	**17.98**	**61.12**	16.93	**6.70**	**80.92**
AugMix on Honda100k	-11.41	63.85	14.08	70.64	**68.95**	47.69	40.12	**61.68**	16.32	88.36
Ours on Honda100k	**-2.55**	**67.35**	**19.88**	**53.26**	65.10	**48.60**	**36.94**	51.90	**18.29**	**72.84**
AugMix on Waymo	18.27	45.40	23.30	59.31	**22.95**	16.92	66.36	**57.65**	29.10	55.63
Ours on Waymo	**20.34**	**46.85**	**26.76**	**52.84**	21.34	**18.24**	**64.58**	56.98	**31.12**	**53.18**

Table 2.3: Performance improvement of different datasets with different sizes against the baseline using the Nvidia backbone. Our method outperforms AugMix considerably in most cases.

3) up to 33%. Lastly, when tested on unseen factors (Scenario 4), our algorithm maintains the best performance by 34% in MMAI, while reducing mCE to 62%.

Compared to AugMix [9], our adversarial approach can select the most challenging datasets for training, thus improving model robustness. MaxUp [57] selects only the worst case among all augmentation data, which may lead to the loss of data diversity. In contrast, our method selects the worst cases in *all* perturbation types (i.e., one dataset per factor), thus improving generalizability. Compared to [69], which performs the adversarial process in feature space by perturbing fea-

ture statistics, our method is able to utilize vast prior information generated by sensitivity analysis to reduce the search space. Lastly, compared to the basic data augmentation method, which uses all generated data in training, our method selects the most useful data for training, and thus improves computational efficiency while minimizing data generation.

Comparison with different backbones. We test three backbones: Nvidia network [58], Comma.ai network [70], and ResNet152 [71], on the Honda dataset. The results shown in Table 2.2 indicate that our method outperforms AugMix in most cases. In general, our method achieves better performance on shallow networks than deep networks. But even on very deep networks such as ResNet152, our method achieves more than 5% improvement in all cases, except Scenario 1.

Comparison on different datasets. To demonstrate that our method does not overfit a particular dataset, we experiment four independent datasets: Audi [59], Honda [60], SullyChen [24], and Waymo [25]. We use the Nvidia network as the backbone for these experiments. Table 2.3 shows that our method achieves consistently better performance across all four datasets. Furthermore, our method obtains up to **97%, 87%, 77%** accuracy improvement on the single, combined, unseen perturbations, while achieving **90%, 74%, 61%** relative error reduction in some cases, respectively.

Comparison on efficiency. To analyze the efficiency of our method, we visualize the relationship between *training time vs. robustness* in Fig. 2.4, using the Nvidia backbone [58] and SullyChen dataset [24]. The x-axis is training time (in seconds) and the y-axis represents the overall robustness, i.e., the average accuracy

	Standard	Cutout	Mixup	CutMix	AutoAugment*	Adv Training	AUGMIX	Ours
AllConvNet	56.4	56.8	53.4	56.0	55.1	56.0	42.7	**25.6**
DenseNet	59.3	59.6	55.4	59.2	53.9	55.2	39.6	**26.3**
WideResNet	53.3	53.5	50.4	52.9	49.6	55.1	35.9	**20.5**
ResNeXt	53.4	54.6	51.4	54.1	51.3	54.4	34.9	**15.4**
Mean	55.6	56.1	52.6	55.5	52.5	55.2	38.3	**22.0**

Table 2.4: Average classification error (in %) across several architectures. Our method is able to reduce mean corruption errors than the previous SOTA methods by 16.3%-34.1% on CIFAR-100-C data.

of the 4 test scenarios (clean, single, combined, and unseen perturbations). Our method outperforms other methods with higher accuracy and efficiency, e.g., after 1/10 the training time (2,500 sec), our performance is already better than others' final performance (with 25,000 sec of training time). Our method is more efficient than others due to discretization of the augmentation space and selection of the most effective augmentation data during the sensitivity analysis. Furthermore, we select the hardest data during training to enable the system to learn more efficiently. Notice that although the frequency space increases the training time per epoch ($>$ 2x longer, mainly in the *Fast Fourier Transform* process), it can help to obtain better performance after 7,000 sec even with the same amount of training time, compared with our approach without the frequency-space method.

Performance on other image processing tasks. To show the generalizability of our method, we test it on classification using CIFAR-100. For a fair comparison, we use the same setting as of AugMix (i.e., same perturbations, training and testing data, backbone, and code). As shown in Table 2.4, our method consistently outperforms all backbones, reducing **16.3%** to **34.1%** in mean errors. The data for other methods come from the AugMix paper [9]. We also test our

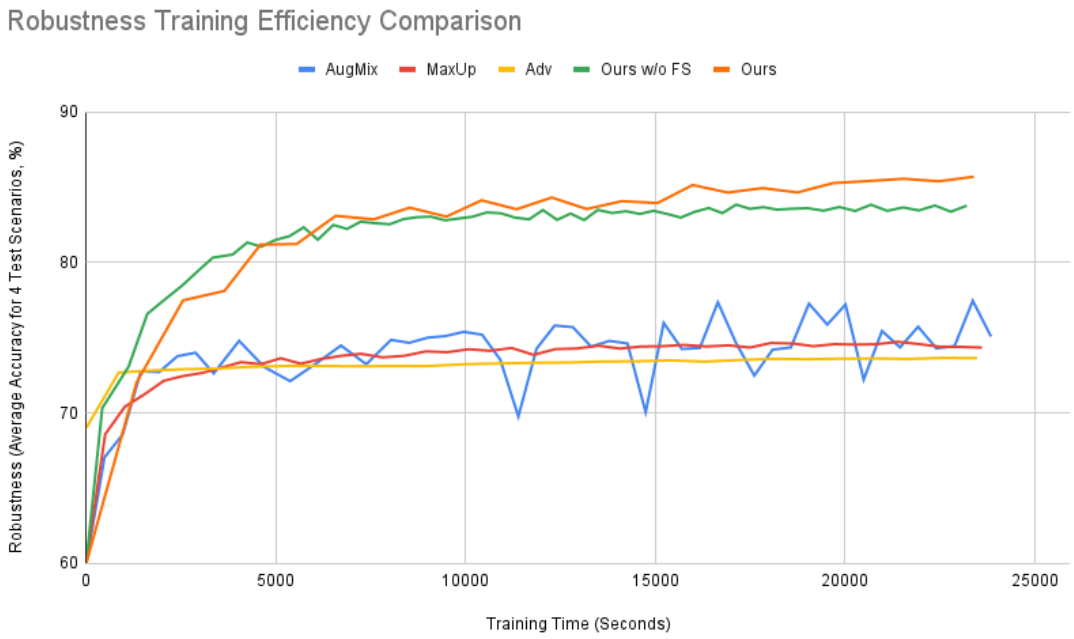


Figure 2.4: Efficiency comparison between our method and other SOTA methods with the Nvidia Network [58] on the SullyChen dataset [24]. The x-axis is training time (in seconds), while the y-axis displays overall robustness, i.e., the average accuracy of the 4 test scenarios (clean, single perturbation, combined perturbation and unseen perturbation). Our method outperforms other methods with higher accuracy and better efficiency. For example, after 1/10 the training time (2500 sec), our performance is already better than others' final performance (after 25,000 sec of training).

method on detection in driving. Specifically, we use the Audi dataset [59] and the Yolov4 network [72] as base settings, and implement ours based on Yolov4. The result shows that our algorithm improves robustness in most scenarios (about 3%-5% mAP on average). See Appendix 2.7.7.

Decoupling Ratio and Generalization Ratio. We propose two new metrics to evaluate how well a training method can allow a model to perform on combined perturbations and unseen perturbations respectively, while only learning on single perturbations one at a time. (**Decoupling Ratio**) Let $MA(D_1, D_2)$ be the

mean accuracy (MA) result of a model trained on dataset D_1 and test on dataset D_2. Let $P_i^*(D)$ be the ith ($i = 1, ..., n$) perturbation taking dataset D as input and producing a perturbed dataset. Let $NP^*(D) = P_n^*(P_{n-1}^*(...P_1^*(D))...)$ (datasets with combined/nested perturbations), $UP^*(D) = \bigcup_{i=1}^{n} P_i^*(D)$ (the union of datasets with single perturbations), D_{tr} be the training set, and D_{te} be the test set, then the decoupling ratio is

$$r_d = \frac{MA(UP^*(D_{tr}), NP^*(D_{te}))}{MA(UP^*(D_{tr}), UP^*(D_{te}))}.$$

$r_d = 1$ means when the model is trained using the union of single-perturbation images, the test performance on combined-perturbation images (not training domain) can be as good as the test performance on the union of single-perturbation images (training domain). Note that different models and training methods may influence the MA function, thus potentially affecting the decoupling ratio.

(**Generalization Ratio**) Keeping the definitions of $MA(D_1, D_2)$, $P_i^*(D)$, D_{tr}, D_{te}, and $UP^*(D)$ as of Decoupling Ratio, let $Q_j^*(D)$ be the jth ($j = 1, ..., m$) perturbation where $P_i^* \neq Q_j^*$, and $UQ^*(D) = \bigcup_{j=1}^{m} Q_j^*(D)$ (the union of datasets with unseen single perturbations), then the generalization ratio is

$$r_g = \frac{MA(UP^*(D_{tr}), UQ^*(D_{te}))}{MA(UP^*(D_{tr}), UP^*(D_{te}))}.$$

$r_g = 1$ means when the model is trained using the union of single-perturbation images, the test performance on unseen-perturbation images (not training domain) can be as good as the test performance on the union of known single-perturbation

images (training domain). Similarly, different models and training methods may influence the MA function, thus potentially affecting the generalization ratio.

Tested on our dataset ($UP^*(D_{tr})$ is the training set with our basis perturbations, $NP^*(D_{te})$ is the data in Scenario 3, and $UQ^*(D_{te})$ is the data in Scenario 4), our method can achieve a high decoupling ratio $r_d = \frac{0.7463}{0.8836} = 0.84$ and a high generalization ratio $r_d = \frac{0.8291}{0.8836} = 0.94$. We exclude these two ratios for other methods because they did not trained on only single perturbations.

Effectiveness visualization. Using the salience map on several combined samples in Fig. 2.8 shown in Appendix 2.7.8, we demonstrate that our method can help the network to focus on important areas (e.g., the road in front) instead of random areas on perturbed images. We also show t-SNE [73] of feature embeddings from the baseline and our method in Fig. 2.9 shown in Appendix 2.7.8. As a result, features from our method are more uniformly distributed, indicating the reduction of the domain gaps created by the perturbations, thus improving robustness.

Benchmarking datasets. We plan to release our perturbed datasets for benchmarking, which will contain augmented datasets from Audi [59], Honda [60], Waymo [25], and SullyChen dataset [24]. Each will include a base dataset and datasets with five levels of perturbation in blur, noise, and distortion, ten levels of variations in the channels R, G, B, H, S, V, multiple combined perturbations over all nine factors using our implementation, and five levels of each unseen simulated factor, including snow, fog, frost, motion blur, zoom blur, pixelate, and jpeg compression using ImageNet-C. In total, there are 480 datasets and about 26M images. The ground-truth steering angles (or angular velocity of the vehicle) for all images

will also be provided for validation, along with the code to generate the perturbed datasets. The parameters for data generation can be found in Appendix 2.7.2.

2.6 Conclusion and Future Work

In this chapter, we first analyze the influence of different image-quality attributes on the performance of the task "learning to steer". We study nine image attributes and find that image degradations due to perturbation on these 9 attributes can impact task performance at various degrees. By using Fréchet Inception Distance (FID) as a unified metric, we conduct sensitivity analysis in the MA-FID space. Leveraging the sensitivity analysis results, we propose an effective and efficient training method to improve the generalization of learning-based steering under various image perturbations. Our model not only improves the task performance on the base dataset, but also achieves significant performance improvement on datasets with a mixture of perturbations (up to 87%), as well as unseen adversarial examples including snow, fog, and frost (up to 77%).

Our method can be easily extended and applied beyond the set of factors and the learning algorithms analyzed in this study. It can also generalize to analyzing any arbitrarily high number of image/input factors, other learning algorithms, and multimodal sensor data. Lastly, other autonomous systems where the *perception-to-control* functionality plays a key role can possibly benefit from our technique as well. We will release the generated datasets for benchmarking the robustness study of learning algorithms for autonomous driving, as well as the code. Our method cur-

rently uses discretization to achieve efficient training, but further optimization for our implementation is possible. Our framework is generalizable to other image factors, learning algorithms, multimodal sensor data, and other perception-to-control tasks.

Acknowledgement: This work was supported in part by ARO SI Program, DARPA GARD program, ONR MURI program, Elizabeth S. Iribe Chair, Barry Mersky, Capital One and Pier G. Perotto E-Nnovate Endowed Professorships.

2.7 Appendix

2.7.1 Dataset Samples

We show different kinds of perturbations in our benchmarks in Fig.2.5. Specifically, our benchmarks include 9 basic types of perturbations, including Gaussian blur, Gaussian noise, radial distortion, and RGB and HSV channels. Another type of datasets include multiple perturbations, where we create multiple random combinations of the basic perturbations. We also include 7 types of previously unseen perturbations (during training) from ImageNet-C [50], which are snow, fog, frost, motion blur, zoom blur, pixelate, and jpeg compression. For each type of perturbation, we generate 5 or 10 levels of varying intensity based on sensitivity analysis in the FID-MA space.

We show more image samples of unseen perturbations in Fig. 2.6.

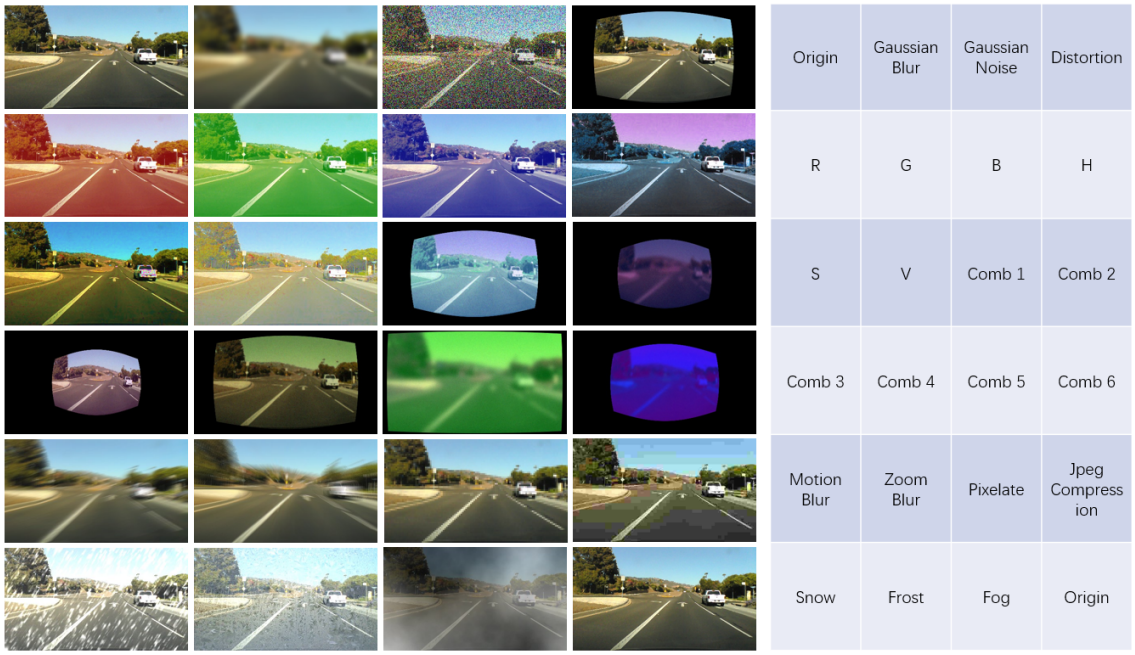


Figure 2.5: Sample images of our benchmark. We show our benchmark has 22 different types of perturbations. Also, we have 10 levels for R, G, B, H, S, V (5 levels in darker and 5 levels in lighter shades), and 5 levels for each of the other types of perturbations.

2.7.2 Perturbed Datasets

We use Gaussian blur [61] (w.r.t standard deviation), additive white Gaussian noise (AWGN) (w.r.t standard deviation), and radial distortion [63] (w.r.t radial distortion parameter $k1, k2$), respectively. For representing channel-level perturbations, we use a linear model: denote the value range of one channel as $[a, b]$, in the darker direction, we set $v_{new} = \alpha a + (1 - \alpha)v$; in the lighter direction, we set $v_{new} = \alpha b + (1 - \alpha)v$. The default values are $a_C = 0$ and $b_C = 255$, where C represents one channel. To exclude a complete dark image, we set $a_V = 10$ and $b_H = 179$.

In our sensitivity analysis experiments, we first select 10 samples for each of

Figure 2.6: Unseen perturbation examples in our experiments. We use "snow", "frost", "fog" (left to right; first row), and "motion blur", "zoom blur", "pixelate", "jpeg compression" (left to right; second row) from the corruptions in ImageNet-C [50].

blur, noise, distortion, channel (R, G, B, H, S, V) darker, and channel lighter, then reduce to $n = 5$. We set $n = 5$ since a smaller number like $n = 2$ will decrease the algorithm performance greatly, while a larger number like $n = 8$ will decrease the efficiency dramatically.

The final representative datasets from the sensitivity analysis and used for improving the generalization of the learning task are introduced in the following.

- R: the base dataset, Audi [59], Honda [60], or SullyChen [24] dataset;
- $B1, B2, B3, B4, B5$: add Gaussian blur to R with standard deviation $\sigma = 1.4$, $\sigma = 2.9$, $\sigma = 5.9$, $\sigma = 10.4$, $\sigma = 16.4$, which are equivalent to using the kernel (7, 7), (17, 17), (37, 37), (67, 67), (107, 107), respectively;
- $N1, N2, N3, N4, N5$: add Gaussian noise to R with $(\mu = 0, \sigma = 20)$, $(\mu = 0, \sigma = 50)$, $(\mu = 0, \sigma = 100)$, $(\mu = 0, \sigma = 150)$, $(\mu = 0, \sigma = 200)$, respectively;
- $D1, D2, D3, D4, D5$: distort R with the radial distortion $(k_1 = 1, k_2 = 1)$, $(k_1 = 10, k_2 = 10)$, $(k_1 = 50, k_2 = 50)$, $(k_1 = 200, k_2 = 200)$, $(k_1 = 500, k_2 =$

500), respectively. k_1 and k_2 are radial distortion parameters, the focal length is 1000, and the principle point is the center of the image.

- $RD1/RL1, RD2/RL2, RD3/RL3, RD4/RL4, RD5/RL5$: modify the red channel of R to darker (D) / lighter (L) values with $\alpha = 0.02$, $\alpha = 0.2$, $\alpha = 0.5$, $\alpha = 0.65$, $\alpha = 1$.

- $GD1/GL1, GD2/GL2, GD3/GL3, GD4/GL4, GD5/GL5$: modify the green channel of R to darker (D) / lighter (L) values with $\alpha = 0.02$, $\alpha = 0.2$, $\alpha = 0.5$, $\alpha = 0.65$, $\alpha = 1$.

- For B, H, S, V channels, we use similar naming conventions for notation as for the red and green channels.

- $Comb1$: $R_\alpha = -0.1180$, $G_\alpha = 0.4343$, $B_\alpha = 0.1445$, $H_\alpha = 0.3040$, $S_\alpha = -0.2600$, $V_\alpha = 0.1816$, $Blur_\sigma = 3$, $Noise_\sigma = 10$, $Distort_k = 17$

- $Comb2$: $R_\alpha = 0.0420$, $G_\alpha = -0.5085$, $B_\alpha = 0.3695$, $H_\alpha = -0.0570$, $S_\alpha = -0.1978$, $V_\alpha = -0.4526$, $Blur_\sigma = 27$, $Noise_\sigma = 7$, $Distort_k = 68$

- $Comb3$: $R_\alpha = 0.1774$, $G_\alpha = -0.1150$, $B_\alpha = 0.1299$, $H_\alpha = -0.0022$, $S_\alpha = -0.2119$, $V_\alpha = -0.0747$, $Blur_\sigma = 1$, $Noise_\sigma = 6$, $Distort_k = 86$

- $Comb4$: $R_\alpha = -0.2599$, $G_\alpha = -0.0166$, $B_\alpha = -0.2702$, $H_\alpha = -0.4273$, $S_\alpha = 0.0238$, $V_\alpha = -0.2321$, $Blur_\sigma = 5$, $Noise_\sigma = 8$, $Distort_k = 8$

- $Comb5$: $R_\alpha = -0.2047$, $G_\alpha = 0.0333$, $B_\alpha = 0.3342$, $H_\alpha = -0.4400$, $S_\alpha = 0.2513$, $V_\alpha = 0.0013$, $Blur_\sigma = 35$, $Noise_\sigma = 6$, $Distort_k = 1$

- $Comb6$: $R_\alpha = -0.6613$, $G_\alpha = -0.0191$, $B_\alpha = 0.3842$, $H_\alpha = 0.3568$, $S_\alpha = 0.5522$, $V_\alpha = 0.0998$, $Blur_\sigma = 21$, $Noise_\sigma = 3$, $Distort_k = 37$

The datasets $Comb1$ through $Comb6$ are generated by randomly sampling parameters of each type of perturbation, e.g. blur, noise, distortion, and RGB and HSV channels, and combining these perturbations together. The parameters listed here are the parameters for the corresponding examples used in the experiment.

We also show the comparison when choosing different basis perturbations in Table. 2.5. The final set we used are better than other sets (e.g., V channel only, or V channel + B channel + Blur) on clean and unseen perturbation scenarios. Notice we don't compare them on single perturbation and combined perturbation scenarios, as the basis perturbations are different.

	Scenarios			
	Clean	Unseen Perturbation		
Basis Perturbations	AMAI↑	MMAI↑	AMAI↑	mCE↓
V channel	-3.33	14.61	1.64	98.54
V channel, B channel, Blur	-0.83	15.77	2.79	95.64
Ours	**2.12**	**33.16**	**13.81**	**61.61**

Table 2.5: Performance of different basis perturbations. The set used is better than other sets on clean and unseen perturbation scenarios. We do not compare them on single perturbation and combined perturbation scenarios, as the basis perturbations are different.

2.7.3 Dataset details

We use Audi dataset [59], Honda dataset [60], Waymo dataset [25], and SullyChen dataset [24]. Among the autonomous driving datasets, Audi dataset is one of the latest dataset (2020), Honda dataset is one of the datasets that have a large

amount of driving videos (over 100+ videos), Waymo dataset includes many environmental conditions such as weather and lightening and is a large dataset (390k frames for perception), and SullyChen dataset is collected specifically for steering task and has a relatively long continuous driving image sequence on a road without branches and has relatively high turning cases.

For Audi dataset [59], we use the "Gaimersheim" package which contains about 15,000 images with about 30 FPS. For efficiency, we adopt a similar approach as in [58] by further downsampling the dataset to 15 FPS to reduce similarities between adjacent frames, keep about 7,500 images and align them with steering labels. For Honda dataset [60], which contains more than 100 videos, we first select 30 videos that are most suitable for learning to steer task, then we extract 11,000 images for Honda10K and 110,000 images for Honda100K from them at 1 FPS, and align them with the steering labels. For SullyChen dataset [24], images are sampled from videos at 30 frames per second (FPS). We then downsample the dataset to 5 FPS. The resulting dataset contains approximately 10,000 images. All of them are then randomly split into training/validation/test data with an approximate ratio 20:1:2. For Waymo dataset [25], we use the data from the perception part directly, which is already split into train/validation/test folders (exclude the domain adaptation data). The training set is about 163k frames, while the test set is about 33k frames. The Waymo dataset doesn't contain steering labels directly, but it contains angular velocity data from IMU. Instead of predicting steering angle, we predict the angular velocity w.r.t. the gravity axis. The only modification is we scale up the angular velocity to meet the range of steering angle, to make it a similar regression problem

as steering regression.

There are several good autonomous driving datasets, but not all of them are suitable for the end-to-end learning to steer task. For example, KITTI [74], Cityscapes [75], OxfordRoboCar [76], Raincouver [77], etc., do not contain steering angle labels. Some well-known simulators like CARLA [29] can generate synthetic datasets, but our work focuses on real-world driving using real images. There are also several other datasets that contain steering angle labels (e.g., nuScenes [78], Ford AV [79], Canadian Adverse Driving Conditions [80], etc), but we didn't use them all because the results on the three datasets we chose can already show the effectiveness of our method.

2.7.4 FID-MA and L2D-MA

We adopt the Fréchet Inception Distance (FID) [64] as a unified metric for our sensitivity analysis (instead of using the parameter values of each image factor) for three reasons. First, given the autonomous driving system is nonlinear, variance-based measures would be more effective for sensitivity analysis of the network. FID can better capture different levels of image qualities than the parameter values of each factor, because the correspondence between the parameter values and image quality of each factor is not linear. Second, using FID, we can map the parameter spaces of all factors into one space to facilitate the sensitivity analysis. Lastly, FID serves as a comprehensive metric to evaluate the distance between two image datasets: image pixels and features, and correlations among images—these mean-

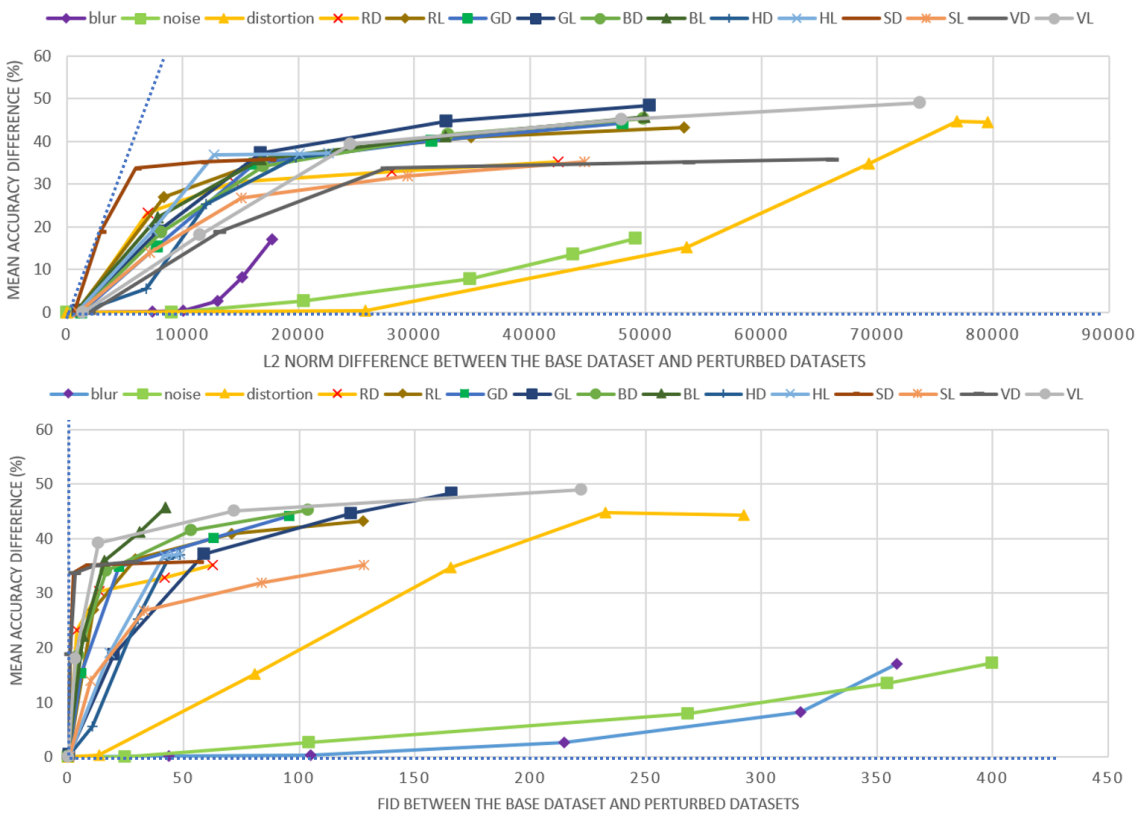


Figure 2.7: The relation between L2 norm distance and MA difference (top), and the relation between FID and MA difference (bottom). The FID space can better capture the difference among various factors affecting image quality better than the L2D space, i.e., the range of the curves' first-order derivative is larger in FID space than in L2D space (see the angle between the two dot lines).

ingful factors to interpret the performance of a learning-based task—are all taken into consideration.

We first empirically select m parameter values for blur, noise and distortion perturbation severity, and $2m$ parameter values for R, G, B, H, S, V (m in the darker direction and m in the lighter direction), generate perturbed datasets using these parameter values, and compute their corresponding MA using the trained model on R (see numeric results in Appendix 2.7.9). At this point, we can obtain the relationship between MA and parameter values for each factor, but **the parameter**

values for each factor are not directly comparable to each other. Notice for each perturbation, one parameter value can generate one perturbed dataset. Thus, we can calculate the FID between this new dataset and clean dataset. In this way, we map the correspondences between the MAs and the parameter values into the FID space, i.e., FID-MA relationship. By leveraging this new FID-MA space, we can minimize the number of parameter samples for each perturbation, while maintaining a similar FID-MA curve between the sampled dataset and the original one to improve the computational efficiency of training (see Sec 2.4.3). **A high-level guideline is, in the range that has high sensitivity (See definition in Sec. 2.4.2) there are denser sample points (closer FID values between sample points), while in the range that has low sensitivity there are sparser sample points**. Specifically, we retain the min and max parameter values, and pick $n-2$ other parameter values to maximize the minimum 'MA and normalized FID difference' between adjacent sample points (here we assume the FID-MA curve is approximately monotonic):

$$\underset{\substack{Q=\{p_{q_1},p_{q_2},...,p_{q_n}\}\subseteq P \\ q_1<q_2<...<q_n}}{\text{minimize}} \underset{i}{\text{minimize}} |MA(p_{q_{i+1}}) - MA(p_{q_i})| + \alpha|FID(p_{q_{i+1}}) - FID(p_{q_i})|$$

where $P = \{p_1, p_2, ..., p_m\}$ is the m parameter values we chose in the beginning, $Q = \{p_{q_1}, p_{q_2}, ..., p_{q_n}\} \subseteq P$ is the n parameter values we want to keep, $MA(p_j)$ and $FID(p_j)$ are the mean accuracy value and FID value related to the parameter value p_j, and α is the normalization parameter which equals to the reciprocal of the largest

FID (within the FID values related to the chosen parameter values). Notice this is not the only criterion to achieve the goal of "sample denser points in high sensitivity range and sample sparser points in low sensitivity range". We also experimented with other criteria, like replacing $|MA(p_{q_{i+1}}) - MA(p_{q_i})| + \alpha|FID(p_{q_{i+1}}) - FID(p_{q_i})|$ with $|MA(p_{q_{i+1}}) - MA(p_{q_i})|$ (MA difference only), which achieves slightly worse results but still works.

We set $m = 10$, try $n = 3, 5, 8$, and find $n = 5$ is the best one ($n = 5$ can lead to similar robustness of the model as $n = 8$ while taking less time, and more robust than $n = 3$). Examples of the resulting images are shown in Fig. 2.2 (more in Appendix 2.7.1). Detailed descriptions of the final perturbed datasets are provided in Appendix 2.7.2. We also provide a detailed analysis of MA differences in FID space in Appendix 2.7.4.

The final MA differences in the FID space are visualized in Fig. 2.7. Since FID aligns different factors into the same space, we can compare the performance loss of all factors at various levels. Notice that the values in the near-zero FID range (i.e., FID$<$ 50) are more commonly found in real-world applications. We first observe that the learning-based steering task is more sensitive to the channel-level perturbations (i.e., R, G, B, H, S, V) than the image-level perturbations (i.e., blur, noise, distortion). Second, the task is least sensitive to blur and noise but most sensitive to the V channel, the intensity value. Third, for the same color channel, darker and lighter levels appear to have different MAs at the same FID values. Compared with other analysis studies on the "learning to steer" task, e.g., [66] and [67], our method is the first to transfer perturbations from multiple parameter

spaces into one unified space to enable the cross-factor comparison, e.g., the task is more sensitive to the V-channel perturbation than perturbations in other attributes.

We illustrate the relationship between FID and Mean Accuracy (MA) Difference, and the relationship between L2 norm distance (L2D) and Mean Accuracy (MA) Difference in Fig. 2.7. As shown in the figure, the FID space can better capture the difference among various factors affecting image quality better than the L2D space, i.e., the range of the curves' first-order derivative is larger in FID space than in L2D space (see the angle between the two dot lines).

2.7.5 Frequency Branch of Our Method

When there are frequency-related perturbations (e.g., using Gaussian noise is increasing high frequency component of the image, while using blur operation will reduce high frequency component), a frequency branch can be added to our architecture. Formally, we do a standard 2-D Fourier Transform, and using the absolute value of each complex number and form another channel of the image, thus the input image of the network has 4 channels. The transform equation is:

$$Y_{p,q} = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \omega_m^{jp} \omega_n^{kq} X_{j,k}$$

where ω_m and ω_n are complex roots of unity:

$$\omega_m = e^{-2\pi i/m}$$

$$\omega_n = e^{-2\pi i/n}$$

i is the imaginary unit. p and j are indices that run from 0 to $m-1$, and q and k are indices that run from 0 to $n-1$.

Notice this frequency branch is optional, our method can already outperform others without this branch. It can help improve the accuracy, but also has limitation: about 1.5x training time. Thus this is an option depends on whether users want the model to be more accurate or more efficient.

2.7.6 Second Stage of Our Method

As introduced in Sec. 2.4.3, the second stage is designed to boost clean accuracy, where the model is fine-tuned solely on clean data. To meanwhile maintain the performance on the perturbed data, we terminate this stage if the overall validation accuracy on clean and perturbed data decreases. Otherwise, this stage continues until it reaches the maximum number of iteration or the MA loss decreases to our expected threshold. In most cases, the network can already learn well the first stage on both clean and perturbed data, thus it will converge very fast in the second stage and do early break without influencing the performance or increasing the training time, as shown in the first row of Table. 2.6. But we found in some cases, the first stage can make the network learn well on perturbed data, but can not perform on clean data as well as a network trained on only clean data. In this case, adding the second stage can help the network perform better, while doesn't reduce the performance on perturbed data too much. Moreover, our method can perform better than AugMix even without the second stage. See Table. 2.6 for details.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
AugMix on Audi	-8.24	81.89	32.22	55.27	75.49	50.23	41.98	73.06	27.39	77.51
Ours on Audi w/o 2	5.86	94.95	47.78	11.83	**88.42**	60.31	27.18	75.16	**32.91**	**39.04**
Ours on Audi	**5.98**	**97.57**	**48.50**	**10.27**	87.56	**62.38**	**25.80**	**77.22**	32.71	39.14
AugMix on Honda100k	-11.41	63.85	14.08	70.64	68.95	47.69	40.12	**61.68**	16.32	88.36
Ours on Honda100k w/o 2	-7.40	66.69	17.77	58.31	**69.98**	47.43	37.88	58.27	17.64	80.50
Ours on Honda100k	**-2.55**	**67.35**	**19.88**	**53.26**	65.10	**48.60**	**36.94**	51.90	**18.29**	**72.84**

Table 2.6: Ablation study for the second stage. "w/o 2" stands for "without the second stage training". On Audi dataset, the network is already well trained on both clean and perturbed data, thus the second stage won't make great differences. But on Honda100k dataset, the performance on the clean dataset for the first stage is not well, and adding the second stage can improve the performance on clean data while doesn't influence performance on perturbed data too much. But even without the second stage, our method can perform better than AugMix.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AmAPI↑	MmAPI↑	AmAPI↑	mCE↓	MmAPI↑	AmAPI↑	mCE↓	MmAPI↑	AmAPI↑	mCE↓
AugMix	-2.23	10.63	1.54	97.35	3.84	1.12	96.18	3.06	1.95	98.74
Our method	**-1.12**	**16.21**	**3.40**	**95.72**	**7.53**	**4.94**	**94.92**	**5.88**	**2.93**	**96.86**

Table 2.7: Performance comparison for detection task against the baseline performance [72]. Our method outperforms the AugMix in all cases, with about 1%-3% mAP improvement on average, while reducing the mCE by 1%-2%.

2.7.7 Performance on Detection

We also test our algorithm on the detection task in autonomous driving. We use the Audi dataset [59] (3D Bounding Boxes) and the Yolov4 network [72] as base settings, and then implement our algorithm based on Yolov4. Table 2.7 shows that our algorithm also improves the model robustness in most scenarios (about 3%-5% mAP improvement on average), and is consistently better than AugMix (about 1%-3% mAP improvement on average).

2.7.8 Visualization

We show the saliency map visualization in Fig. 2.8. Our method can help the network to focus on important areas (e.g., the road in front) instead of random areas on perturbed images.

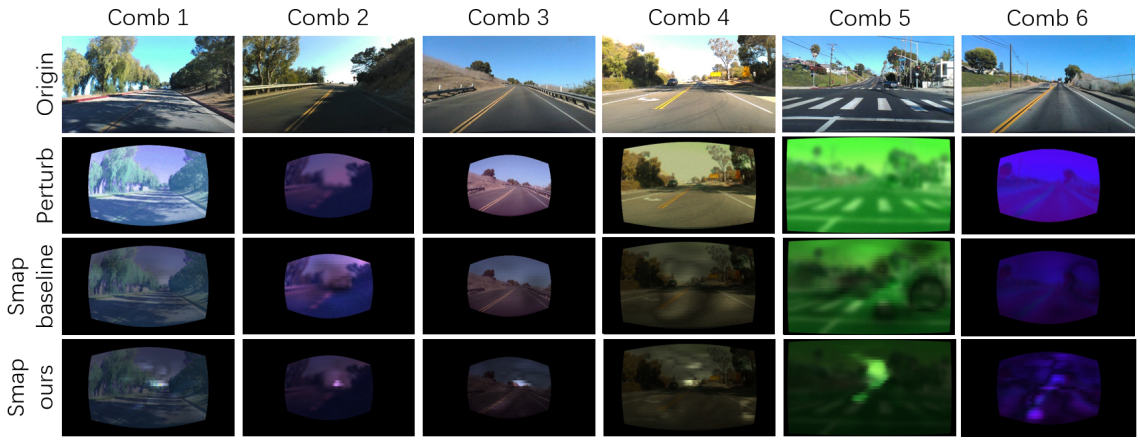


Figure 2.8: Saliency map samples using the baseline method and our method, where the model is tested on different combinations of perturbations shown as columns. We show the original image, perturbed image with a chosen effect, saliency map of the baseline model, and saliency map of our method from top to bottom rows. Using our method, the network focuses more on the important areas (e.g., road in front) instead of random areas on the perturbed images.

We also show the t-SNE [73] visualization of feature embeddings for the baseline method and our method in Fig. 2.9. The features from the baseline method are more clustered by color (e.g., the left circle in the left image mainly contains red dots, and the right circle in the left image mainly contains yellow dots), indicating there are domain gaps between the perturbed data and original data; while the features from our method are more uniformly distributed, suggesting that our method is able to reduce the domain gaps from perturbations, i.e., improve the robustness.

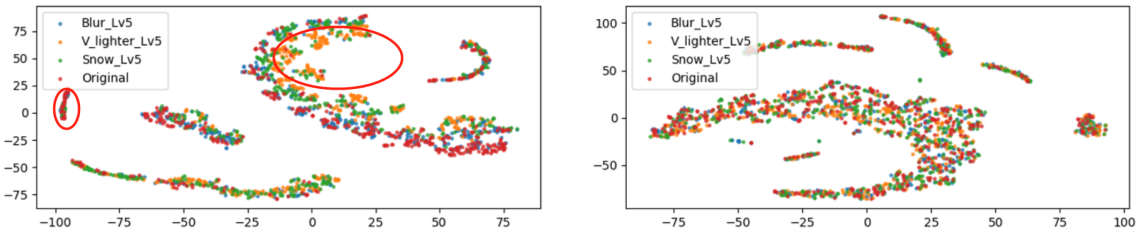


Figure 2.9: t-SNE [73] visualization for features achieved from networks trained by baseline method (left) and our method (right). The features from the baseline method are more clustered by color (e.g., the left circle in the left image mainly contains red dots, and the right circle in the left image mainly contains yellow dots), indicating there are domain gaps between the perturbed data and original data, while the features from our method are more uniformly distributed, suggesting that our method is able to reduce the domain gaps due to perturbations, i.e., improving the robustness.

2.7.9 Experiment data

To quantify our results, we collected mean accuracy (MA) measurements from each experiment, for each pairwise factor and level across methods. Table 2.8 shows the mean accuracy measurements for blur, noise, and distortion factors. The same is of table 2.9, where mean accuracy is measured across levels of RGB or HSV color channels, where each channel serves as a single corruption factor. Table 2.10 presents the MA measurements for scenarios with a combination of factors, and Table 2.11 presents the MA measurements for scenes with previously unseen factors.

Method	Factor	L1	L2	L3	L4	L5
baseline	blur	88.2	88.1	86.1	81.2	73.3
	noise	88.3	86.0	81.4	76.4	73.2
	distortion	88.6	75.0	57.7	**48.8**	49.2
ours	blur	90.6	90.6	90.3	90.7	88.6
	noise	89.7	89.2	86.1	84.2	79.4
	distortion	90.3	89.8	87.0	**84.2**	83.3

Table 2.8: Mean Accuracy of training (in %) using the baseline model and ours, tested on datasets with different levels of blur, noise, and distortion. Levels range from L1 to L5. We achieve up to 35.4% in performance gain (see bold number pair).

Method	Factor	DL5	DL4	DL3	DL2	DL1	LL1	LL2	LL3	LL4	LL5
baseline	R	53.2	55.4	57.9	65.1	87.8	87.7	61.4	52.1	47.4	45.1
	G	44.2	48.2	53.5	73.0	88.5	87.9	69.6	51.2	43.7	**40.0**
	B	43.0	46.8	54.3	69.7	88.2	87.7	66.2	52.5	47.1	42.6
	H	51.3	52.1	63.1	82.8	88.1	88.2	69.3	51.5	51.3	51.2
	S	58.4	63.8	72.6	83.9	88.1	88.3	74.5	61.6	56.5	53.2
	V	52.6	53.2	54.6	69.4	88.5	88.4	70.4	49.1	43.2	39.4
ours	R	88.6	90.1	90.6	90.5	90.5	90.4	90.6	90.6	90.2	89.4
	G	90.0	90.6	90.6	90.5	90.5	90.4	90.5	90.6	90.3	**89.9**
	B	89.1	90.0	90.5	90.4	90.3	90.4	90.4	90.6	90.0	89.3
	H	89.7	90.2	90.2	90.4	90.4	90.7	90.5	90.0	89.2	89.7
	S	88.9	90.0	90.4	90.5	90.6	90.6	90.7	90.7	89.7	86.9
	V	87.8	89.5	90.7	90.8	90.7	90.5	90.6	90.0	84.4	76.4

Table 2.9: Mean accuracy (MA) of training (in %) using the baseline model and ours, tested on datasets with different levels of R, G, B, and H, S, V channel values. DL denotes "darker level", which indicates a level in the darker direction of the channel, while LL indicates "lighter level", which indicates the lighter direction, on levels 1 to 5. We achieve up to **49.9%** in performance gain (see bold number pair).

Method	Comb1	Comb2	Comb3	Comb4	Comb5	Comb6
baseline	59.7	54.0	40.9	**50.0**	54.0	56.3
ours	73.4	68.6	70.9	**83.3**	86.2	65.3

Table 2.10: Mean accuracy (MA) of training (in %) using the baseline model and ours, tested on datasets with several perturbations combined together, including blur, noise, distortion, RGB, and HSV. We achieve up to **33.3%** in performance gain (see bold number pair).

Method	Unseen Factors	L1	L2	L3	L4	L5
baseline	motion_blur	76.4	69.7	62.6	61.1	60.3
	zoom_blur	85.6	83.7	81.8	80.0	78.2
	pixelate	88.2	88.2	88.0	88.3	88.1
	jpeg_comp	88.4	88.0	87.4	85.4	82.2
	snow	62.8	**50.7**	54.9	55.5	55.3
	frost	55.8	52.1	51.7	51.7	51.2
	fog	58.7	55.0	52.4	50.8	48.1
ours	motion_blur	88.0	86.9	84.7	81.4	78.5
	zoom_blur	88.3	86.9	85.5	84.1	82.5
	pixelate	90.3	90.3	89.9	90.5	90.6
	jpeg_comp	90.1	90.2	90.0	89.5	90.0
	snow	87.1	**83.8**	82.0	77.4	76.7
	frost	86.0	83.9	81.1	81.7	80.1
	fog	77.4	71.6	65.6	61.5	56.1

Table 2.11: Mean accuracy (MA) of training (in %) using the baseline model and ours, tested on datasets with previously unseen perturbations at 5 different levels. These types of unseen perturbations do not appear in the training data, and include motion blur, zoom blur, pixelate, jpeg compression loss, snow, frost, and fog, on intensity levels L1 to L5. We achieve up to **33.1%** in performance gain (see bold number pair).

Chapter 3: Auxiliary Modality Learning with Generalized Curriculum Distillation

3.1 Introduction

Learning from images and videos is among some of the most popular research focuses [81, 82, 83], as RGB images are informative and easy to acquire. In addition, RGB camera is cheap and can be easily deployed. There are also works considering multiple modalities, i.e., multi-modal learning [46, 84, 85]. Furthermore, some works consider to use multiple modalities in training but use fewer modalities during test since in certain applications it's difficult to use all modalities during inference. For example, it is expensive to deploy Lidar on commodity self-driving cars, but it's reasonable to equip a few developer's cars with Lidar for training. However, this specific type of learning task, i.e., "test with fewer modalities than during training", is not standardized yet. For example, there is no formal term or definition. There have been concepts, such as "learning with side information" [17], "learning with privileged information" [18], "learning with auxiliary modality" [19], "learning with partial-modalities" [20], and "modality distillation" [21], etc. We therefore formalize these learning tasks as *Auxiliary Modality Learning (AML)* in Sec. 3.3.

To apply AML to real-world tasks, there are some key issues: "*what types of auxiliary modalities can be used, and how to add the auxiliary modalities into the network and make them most effective?*" We systematically list and classify auxiliary modalities in visual computing and network architectures for AML, and then conduct experiments to address these questions. Specifically, we classify the auxiliary modalities into 3 types: low-level sensing data (Type 1), middle-level equivalent representation (Type 2), and high-level conceptual information (Type 3) in Sec. 3.4.1.1, according to the types of information. We also classify the network architectures into four types, according to the mechanism that introduces the auxiliary modality. They are auxiliary modalities in the input (Type A), in the middle (Type B), in the end (Type C), and in the teacher (Type D), as defined in Sec. 3.4.1.2. In addition, we design experiments to see which architecture and which auxiliary modality perform best within each task and across tasks (Sec. 3.4.1.3) that provides experimental guidelines and theoretical foundation to our method in Sec. 3.5.

Given this formal framing, we can apply AML to real-world tasks. There remains the question of explainability: "*Why AML can work without auxiliary modality in the test?*" It's not obvious that adding auxiliary modality only in training can always help improve test performance with only the main modality. In Sec. 3.4.2, we explore this line of inquiries from optimization and data perspectives. Specifically, we introduce a new concept of "supermodel" to support our claim, which also offers insights and inspiration to design the new AML method presented in Sec. 3.5.2.

Based on the detailed analysis in Sec. 3.4, we propose a simple yet effective method, *Smart Auxiliary Modality Distillation (SAMD)*, that can smartly choose

the best auxiliary modality and perform a special auxiliary modality distillation with generalized curriculum distillation. Firstly, in Sec. 3.4.1.3, we show that different auxiliary modalities can contribute to different tasks at a different level, thus we propose a method to choose the best auxiliary modality and estimate upper-bound performance for a given task before actually executing AML. Inspired by *Squeeze-and-Excitation Network (SENet)* [86], we use channel-level attention in the SE block to estimate the AML performance for each modality, and show the consistently positive correlation between them through experiments on different tasks and auxiliary modalities. See details in Sec. 3.5.1. Also, in Sec. 3.4.1.3, we show the knowledge distillation based architecture (Type D) is better than other forms. However, when analyzing the reason for the effectiveness of AML in Sec. 3.4.2, we find the "supermodel condition", which helps AML to perform better, is not fully utilized in the general knowledge distillation based architecture. We thus introduce a new method that uses supermodel in a more effective way that allows the teacher network to be aware of the student's status in a curriculum way, leading to a better distillation. Our method achieves better performance compared to other SOTA methods (Sec. 3.5.2).

Our analysis provides experimental understanding and theoretical underpinning for the simple yet effective method design. To the best of our knowledge, this is the first detailed analysis to guide the design and choices of AML methods for visual computing based on tasks, datasets, and network architectures. In summary, our contributions are:

- Systematically list and classify different types of auxiliary modalities and architectures (Sec. 3.4.1.2) for AML, and analyze the performance behavior of different types of auxiliary modalities and architectures for AML across different datasets, backbones and tasks (Sec. 3.4.1.3). We find (1) architecture effectiveness is relatively consistent across different tasks, datasets and backbones; (2) auxiliary modality effectiveness is consistent within one task with different datasets and backbones, but not consistent across tasks.

- Propose a novel AML method, "Smart Auxiliary Modality Distillation (SAMD)", that automatically (1) chooses the best auxiliary modality for the main distillation process, and (2) performs knowledge distillation under a special "supermodel condition" to enable the teacher network to be aware of the student's status. SAMD achieves SOTA results on variant tasks, with improvement up to **10%** on end-to-end steering task, **5%** on multi-view handwriting classification task, and up to **15.6%** across tasks, etc. (Sec. 3.5).

- Analyze and explain the reasons for the effectiveness of AML from both optimization perspective and data perspective (Sec. 3.4.2), providing theoretical support to the SAMD method.

3.2 Related Work

Auxiliary Modality Learning aims to use auxiliary modality in training to boost the test performance without the auxiliary modality during inference. Cross-modality Learning and Knowledge Distillation are comparatively promising solu-

tions and we discuss related works in each here. More related works are discussed in Appendix 3.8.8.

3.2.1 Cross-modality Learning

To utilize the prior knowledge between different modalities, Gupta et al. [87] learned the representation of one modality with a pretrained network on another modality. Hoffman et al. [17] presented early work on modality hallucination, which used a hallucination network with RGB image as input but tried to mimic a depth network, by combining with RGB network to achieve multimodal learning. Some [18, 21] train the hallucination network with a different process to achieve better performance, while others [19, 20] use GAN or U-Net to generate another paired modality data with one modality. MSD [88] transfers knowledge from a teacher on multimodal tasks by learning the teacher's behavior within each modality. A recent work [89] trains the different modality data in different pipelines and distills the best modality pipeline knowledge to other modality pipelines. In addition to action recognition, AML has also been applied in medical image processing [90, 91]. Specifically, Zheng et al. [92] investigated the effectiveness of shape priors learned from a different modality (e.g., CT) to improve the segmentation accuracy on the target modality (e.g., MRI). Valindria et al. [93] proposed dual-stream encoder-decoder framework, which assigns each modality with a specific branch and extracts cross-modality features with carefully designed parameter sharing strategies. Li et al. [91] exploited the priors of assisted modality to promote the performance on another modality by

enhancing model generalization ability, where only target-modality data is required in the test.

3.2.2 Knowledge Distillation

Knowledge Distillation can be classified as one-way or mutual-learning knowledge distillation. One-way knowledge distillation mainly distills the knowledge of a fixed teacher model (usually large) to a student model (usually small). In the early days, Hinton et al. [94] proposed compressing the knowledge in an ensemble of multiple models into a single model that is much easier to deploy by mimicking the class distribution via softened softmax from the ensemble teacher. Some studies [95, 96] went further to explore the trade-off between the supervision of soft logits and hard task label. Furthermore, there are also methods exploiting the intermediate feature [97, 98, 99] as transferred knowledge, which can improve the middle layer's representational ability in a student network. Other than the one-way distillation from teacher to student, some focus on mutual knowledge distillation among models trained from scratch. This line of research is especially notable for scenarios without an available pretrained teacher model. A significant work is deep mutual learning (DML) [100]. During the training phase, DML uses a pool of randomly initialized models as the student pool, and every student is guided by the output of other students and the task label. [101] go a further step and introduce an anchor model to delimit a subspace within the full solution space of the target problem, which can help to ease the distillation difficulty.

We share a similar philosophy with distillation, but aim to design a cross-modality learning framework to utilize the hidden information from auxiliary modalities, resulting in a different methodology.

3.3 Auxiliary Modality Learning

Auxiliary modality learning offers promising potential, but has not been fully examined. Previous works studied auxiliary modality learning in certain applications without a formal definition or a unified terminology. [17] named this process "learning with side information", [18] called it "learning with privileged information", [19] referred to it as "learning with auxiliary modality", [20] suggested "learning with partial-modalities", [21] introduced the term "modality distillation", etc. In this chapter, we formally define the *Auxiliary Modality Learning (AML)* as follows:

Definition 3.3.1. *Given data with one set of modalities I_M and data with another set of modalities I_A, if a model $\mathcal{M}$ can take both I_M and I_A as input during training, but only use I_M during test, then we call model $\mathcal{M}$ an auxiliary modality model, I_M as the main modality data and I_A as the auxiliary modality data. Furthermore, we call the training process of an auxiliary modality model as auxiliary modality learning.*

Formally, the training process is $\text{minimize}_\theta\, L(\mathcal{M}_\theta, (I_M, I_A), GT)$*, where θ is the weights of the model, L is the loss function, and GT is the ground truth, while the test process is $\mathcal{M}_\theta(I_M)$.*

The goal of AML is to achieve better performance with the help of auxiliary modalities I_A than only train on main modalities I_M. AML can be found in the real world, e.g., when you cannot solve a problem in class, the teacher gives you some hints so the students can understand the relationship between the problem and the answer better. Then, after class, the student can solve similar types of new problems without hints. In this chapter, for visual computing, we fix the main modality as RGB images, but the auxiliary modality can be others, like point cloud, depth map, or other customized formats.

AML is useful in the following scenarios: (1) Getting the extra modality data during test is not feasible. For example, the extra modality can be the human-labeled attention map, which is achievable during training, but we cannot ask the user to label the attention map in real time. (2) Getting the extra modality data during test is feasible but expensive. For example, in autonomous driving, we need to use Lidar to get point cloud data. Using point clouds during training only requires several Lidar sensors on the cars for development, but using point clouds during test means every car needs to install Lidar, which is costly. AML can reduce the cost dramatically compared with the solutions that require the Lidar+camera, and can perform better than the solutions that only use camera. Similarly for robot navigation.

3.4 Analysis

In this section, we aim to do analysis on two key problems of AML when applying on real-world tasks: (1) What kinds of auxiliary modalities can we use, and how can we add them into the network to make them effective? (2) Why AML can work without auxiliary modality in test?

3.4.1 Auxiliary Modality and Architecture

In this section, we first systematically list and classify the types of auxiliary modalities and the types of auxiliary modality learning architecture, then analyze how different auxiliary modalities and architectures can affect auxiliary modality learning through experiments.

3.4.1.1 Types of Auxiliary Modality

Previous auxiliary modality learning works usually only consider one or several given types of auxiliary modality without systematic analysis [17, 18, 19, 20, 21, 87, 88]. This is usually because of the limitation of data sources, e.g., limited sensor types. However, there is actually a wide range of auxiliary modality options that can be used. Except for the sensing data directly from the sensors (like depth map or infra-red image), other data generated from the original image (like segmentation image or frequency image) or even annotated by a human expert (like attention map) can also be used as an auxiliary modality. In our work, we classify the potentially useful auxiliary modalities that are commonly seen in daily life and show their

effectiveness through experiments.

Formally, we suggest the following three types of data, which can be used as auxiliary modality in visual computing, according to the information contained in them:

Type 1: Low-level sensing data with additional information. For example, given the main modality is RGB image, depth map or infra-red image can be used as an auxiliary modality, which is already commonly used [14, 20]. The additional depth or infra-red information can be used when the RGB image is not able to capture enough information like at night.

Type 2: Middle-level representation with equivalent information but in *different spaces*. For example, RGB image can be transferred to/from frequency space with 2D FFT (a one-to-one mapping). Although they contain the same information, one presentation in one space may have a closer relation to the goal, helping the network to learn better.

Type 3: High-level conceptual data with compacted information. For example, expert annotated image with emphasized key features (like attention image). This kind of auxiliary modality helps reduce the redundant noises and helps the network focus on key elements quickly.

Type 1 is the most common type of auxiliary modality, but Type 2 and 3 are also auxiliary modalities that can potentially contribute to the task. See example tasks for different modalities in Appendix 3.8.1.

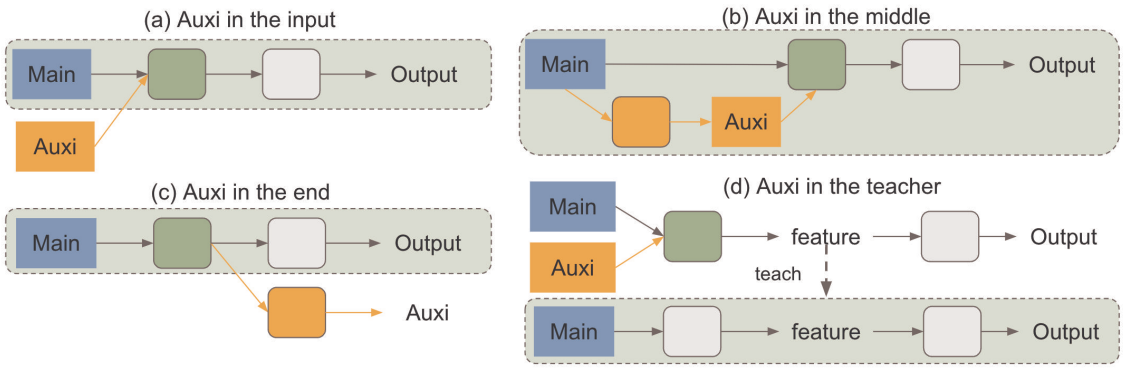


Figure 3.1: Architectures for auxiliary modality learning. Type A: Auxiliary modality in the input. Type B: Auxiliary modality in the middle. Type C: Auxiliary modality in the end. Type D: Auxiliary modality in the teacher network. The dashed area in each type is the test pipeline that only use main modality. See Sec. 3.4.1.2.

3.4.1.2 Architectures for AML

Existing works explore variant ways to achieve the goal of auxiliary modality learning. However, to the best of our knowledge, no one compares architectures in a systematic way [17, 18, 19, 20, 21, 87, 88]. In this section, we list and compare the existing architecture designs for the auxiliary modality learning systematically.

We classify the possible auxiliary modality learning architectures into four types:

Type A: Auxiliary modality in the input, *same architecture as multi-modality learning during training, but only use the main modality branch for test*, as shown in Fig. 3.1(a). General multi-modality architecture is already been studied [14], but multi-modality based AML still needs to be explored.

Type B: Auxiliary modality in the middle as supervision. The basic idea is to generate auxiliary modality with the main modality first, and then use the multi-modality architecture, as shown in Fig. 3.1(b). Existing works like [20, 91, 102]

show the effectiveness of this type of solution.

Type C: Auxiliary modality in the end as supervision, *same architecture as multi-task learning [103] or indirect supervision [104], but only need to use the original task pipeline for test*, as shown in Fig. 3.1(c). The basic idea is the original task and the auxiliary modality generation task share certain common features, thus the auxiliary modality can help the learning of the original task.

Type D: Auxiliary modality in a teacher network and *teach a student network without auxiliary modality, refer to cross-modality knowledge distillation*, as shown in Fig. 3.1(d). Existing works like [21, 89] show the effectiveness of this type of solution.

Notice existing works mostly focus on Type B and D, but few discuss or conduct experiments with Type A and C, which are also potential solutions.

3.4.1.3 Experiments

We conduct experiments to see how different auxiliary modalities and architectures of AML perform within single task and across tasks.

Single Task In this experiment, we consider four factors, auxiliary modality, architectures, backbones and datasets. Our goal is to explore whether there exist general rules under different settings for broader applicability. Different datasets have different properties, e.g. data distribution and data size, while different backbones consist of varying model types and model complexity. We design experiments to answer: (1) Given fixed dataset and backbone, do all the architectures help aux-

iliary modality learning? What's the order among them w.r.t. performance improvement? (2) Given fixed datasets and backbones, do all the auxiliary modalities help auxiliary modality learning? What's the order among them w.r.t. performance improvement? (3) Are the previous two answers consistent across different datasets and backbones?

The experiment setting is described in Appendix 3.8.1. In Table 3.5 (Appendix 3.8.2), we show performance comparison (Mean Accuracy %) with a combination of three auxiliary modalities, four architectures, two backbones, and two datasets. Within this task, we observe:

(1) Knowledge distillation based architecture (Type D) perform best, followed by generation based architecture (Type B). Multi-task based architecture does slightly better than baseline (Type C), while Multi-modality based architecture sometimes hurt the performance (Type A).

(2) All types of auxiliary modality used can help improve performance. Attention image (Type 3) achieves the highest performance improvement, followed by depth map (Type 1). Frequency image (Type 2) only performs slightly better than baseline (the order is proven to be not consistent across tasks in Finding (5) below).

(3) Within this task, the effectiveness order for different auxiliary modalities or architectures are consistent across different datasets or backbones.

Multiple Tasks. However, the rules observed in a single task are not necessarily TRUE across tasks. In this experiment, we consider four factors: auxiliary modality, architectures, backbones and datasets. We design experiments to answer: (4) Is the effectiveness of different architectures consistent across different tasks? (5)

Is the effectiveness of different auxiliary modalities consistent across different tasks?

The experiment setting is presented in Appendix 3.8.1. In Table 3.6 (Appendix 3.8.2), we show performance comparison with a combination of two auxiliary modalities and four architectures across three tasks. Notice when comparing across tasks, we only focus on the relative accuracy order of one task, since the metrics of different tasks are different. We find:

(4) The effectiveness order of different architecture types is consistent for different tasks.

(5) The effectiveness order of different auxiliary modalities may be different for different tasks, but consistent within one task.

Finding (4) supports our choice to design based on architecture Type D (Sec. 3.5.2). Finding (5) motivates the need to select the *best auxiliary modality* at the beginning of a task, but no need to re-select it for using another architecture type, backbone, or dataset (Sec. 3.5.1).

3.4.2 Why AML Works?

We provide experimental results in Sec. 3.4.1.3 to show auxiliary modality learning can work, i.e., although only test with the main modality, using auxiliary modality during training can do better than only using the main modality. This is also supported by other works [17, 18, 19, 20, 21]. However, most of them use experimental results to illustrate their effectiveness, there is no detailed analysis to demonstrate why AML can work. In this section, we explain why AML works from

two perspectives.

3.4.2.1 Optimization Perspective

Here we explain why AML can work from the optimization perspective.

(1) The optimal solution of AML is no worse than learning with the main modality. Inspired by "superset", we first introduce a new concept "supermodel".

Definition 3.4.1. *Given a model $\mathcal{M}_{\theta_A}^{(A)}(I_A)$ (weights θ_A and input I_A), and a model $\mathcal{M}_{\theta_B}^{(B)}(I_B)$ (weights θ_B and input I_B), if for any θ_A, there is a θ_B, s.t. $\mathcal{M}_{\theta_A}^{(A)}(I_A) = \mathcal{M}_{\theta_B}^{(B)}(I_B)$ for any arbitrary valid input data I_A and its superset I_B. Model $\mathcal{M}_B$ is called a* "supermodel" *of $\mathcal{M}_A$.*

See an example of the supermodel in Fig. 4.3. We then introduce a lemma based on the supermodel:

Lemma 3.4.1. *Given a model $\mathcal{M}$ and its supermodel $\mathcal{M}^{(s)}$, the optimal training loss of $\mathcal{M}^{(s)}$ (which is $\arg\min_{\theta^{(s)}} L(\mathcal{M}_{\theta^{(s)}}^{(s)}(I^{(s)}), GT)$) is less than or equal to the optimal training loss of $\mathcal{M}$ (which is $\arg\min_{\theta} L(\mathcal{M}_{\theta}(I), GT)$). where L is the loss function and GT is the ground truth.*

See the proof in Appendix 3.8.4. Now we consider the single network architectures (Type A, B, C in Sec. 3.4.1.2) and the teacher network of Type D, all of them are supermodels of their related main modality pipeline network. Specifically, we can black out the auxiliary modality related branch (e.g., for Type A and C, use the pipeline in the dashed box, for Type B, blackout the auxiliary modality

generation branch, for teacher network in Type D, it's the same as Type 1) by setting the weights of connection layers to specific values (e.g., zeros, depends on the specific type of layer), then the model takes both main and auxiliary modalities will have exactly the same results of the model with only main modality, thus meeting the supermodel definition 4.6.1. According to Lemma. 4.6.1, the AML model is no worse than the original model with only the main modality.

(2) In the case of the same performance, AML allows the optimizer to search in a higher dimension with a higher possibility to find a path learning with the main modality.

Suppose an optimizer g takes model $\mathcal{M}$ and its initial weights θ_0, loss function L, training data I_M as input, and output a path of model weights:

$$g(\mathcal{M}, \theta_0, L, I_M) = \{\theta_0, \theta_1, ..., \theta_{p_1}\} = P_1$$

where $p1$ is the step number, $\theta_{p_1} = \theta^*$ is the optimal solution, and P_1 is the path. Then the AML process on its supermodel $\mathcal{M}^{(s)}$ is

$$g(\mathcal{M}^{(s)}, \theta_0 \oplus \delta_0, L, (I_M, I_A))$$

$$= \{\theta_0 \oplus \delta_0, \theta_1^{'} \oplus \delta_1, ..., \theta_{p_2}^{'} \oplus \delta_{p_2}\} = P2$$

where $\oplus$ is the dimension-level connection, δ as the weights for the auxiliary dimension, $\delta_0 = \delta_{p_2} = 0, \theta_{p_2}^{'} = \theta^*$. This means that only the start and end positions are on the same dimension as the main modality, while in-between it can explore

on a higher dimension (main+auxiliary modality). For any path P_1^*, there is a path P_2^* that represents the same path (by setting $p_2 = p_1$, $\delta_i = 0$ and use $\theta_i^{'} = \theta_i$ for $i = 0, 1, ..., p_1$). But for a P_2^*, there's no P_1^* that can represents the same path when there is a $\delta_i \neq 0$ in P_2^*. It shows even with the same start and end points, the AML can have more path options, which may be easier to be found by a given optimizer, e.g., the blue path in Fig. 3.2 is a gradient descent path in a higher dimension, while the red path in low dimension needs to go uphill in the middle, which is more difficult for the gradient-based optimizer to find solutions.

3.4.2.2 Data Perspective

Next, we explain why AML works from data perspective.

(1) The auxiliary modality can help the main modality training better when main modality data is imbalanced or in shortage. For example, the main modality data has few examples that are the 'hard cases', which lead to a wrong decision boundary. This is common in real-world datasets, e.g., the autonomous driving dataset usually has fewer night data, even worse, has few accident data. After adding the auxiliary modality that provides more information on the hard cases, it would be easier to learn a correct decision boundary, then use this information to guide the training process with the main modality. For example, the infra-red image or depth map contains more information than RGB image when captured at night. This observation explains why the low-level sensing data (Type 1 in Sec. 3.4.1.1) can help AML. See figures in Appendix 3.8.5.

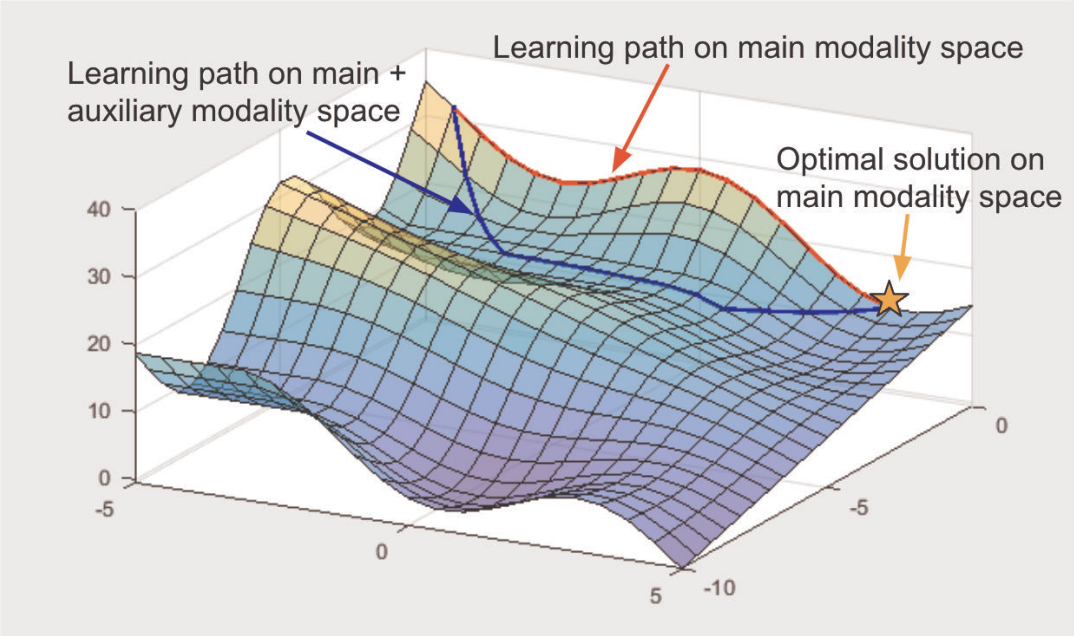


Figure 3.2: "Why AML works" from optimizer perspective. The blue path (AML) is easier to be found by a gradient-based optimizer, since there is no uphill as with the main modality (red).

(2) The auxiliary modality data reveal a simpler mapping function from input to output. As we know, the network is used to learn a mapping function from input to output, e.g., $f(I_M) = y$. However, the function f may be complex and difficult to learn. Then, one solution is to split the complex function f into two parts, i.e., $f(I_M) = f_2(f_1(I_M)) = f_2((I_M, I_A)) = y$, where f_1 is "data reformating function" that contains as much as inductive bias (according to the domain expert experiences) for the given task, thus the f_2 will be simpler than the original f and easier to be learned. This observation explains why middle-level and high-level conceptual data (Type 2 and 3 in Sec. 3.4.1.1) can help AML.

3.5 Smart Auxiliary Modality Distillation (SAMD)

Based on the detailed analysis in Sec. 3.4, we propose a simple yet effective method "Smart Auxiliary Modality Distillation" to choose the best auxiliary modality and do an auxiliary modality distillation.

3.5.1 Auxiliary Modality Choice for a Given Task

As discussed in Sec. 3.4.1.1, there are three types of auxiliary modality that are potentially useful, and each type can have multiple kinds of modalities. Given the conclusion in Sec. 3.4.1.3, there is no consistent best auxiliary modality that can be used for all the tasks, we need to choose the best auxiliary modality that can boost the performance most for a given task. Suppose there are n types of auxiliary modalities, do we need to train n times to find out the best one? The answer is no. In this section, we propose a method that can assist in deciding the importance order for a set of auxiliary modalities within one training process.

Inspired by *Squeeze-and-Excitation Network (SENet)* [86], we use channel-level attention to represent the importance of each modality. Suppose we already have a network f that can take the main modality I_m as input and perform prediction for a given task. Now we have n types of auxiliary modalities that potentially can help. We first pack the different modality data in the channel level, and feed them into the Squeeze-and-Excitation (SE) block [86], followed by a 1x1 convolutional layer to make the channel number to be the same as the main modality I_m, so that the original network f can take that as input and perform prediction. If different

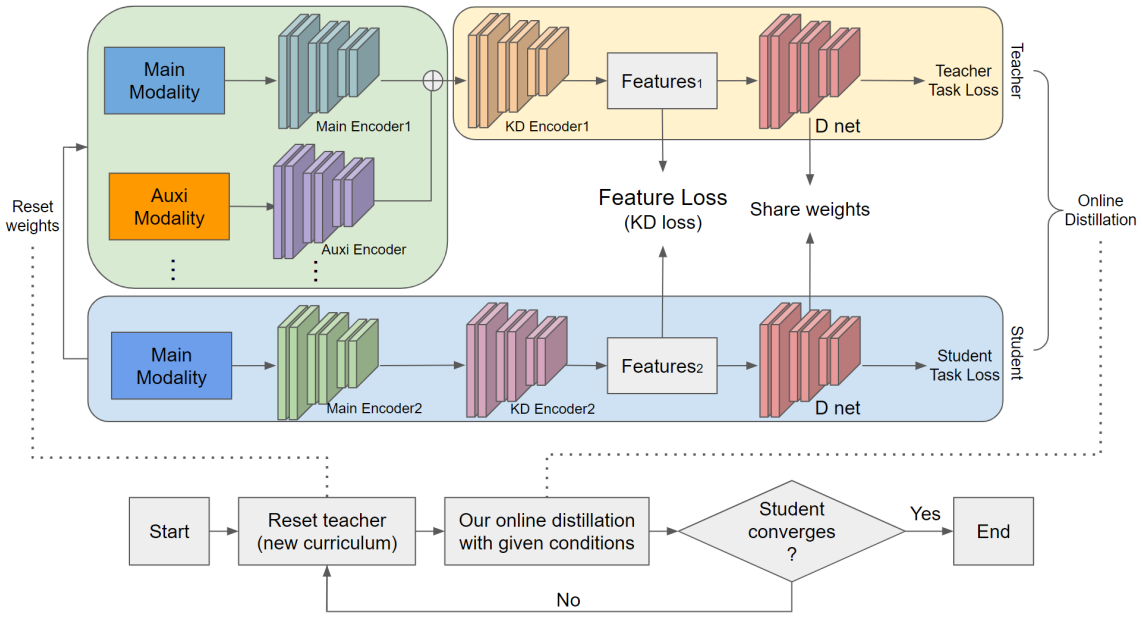


Figure 3.3: SAMD architecture. In each round, a new curriculum learning is started by resetting the teacher weights. Then we train the model with our online distillation, until the student converges. The teacher network should be a supermodel of the student network to enable reset operation, which helps the teacher be aware of the student's status and perform more effectively.

modality data have different image sizes then they should be resized (or add a shallow network to pre-process the data, if necessary) before being packed in the channel level. In our experiments, all the image data with different modalities have the same size, so they can be packed directly. After training the modified network, the channel weights in the SE block can be used to determine the relative importance for the auxiliary modalities, i.e., the modality that has the largest channel weight is the one that can lead to the best AML performance. See Appendix. 3.8.6.

3.5.2 Auxiliary Modality Distillation

Sec. 3.4.1.3 shows the knowledge-distillation (KD) based architecture performs best in most cases. However, when analyzing reasons for the effectiveness of AML in

	Accuracy (%) on various angle threshold τ (degree)				
Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	**mAcc**
[17]	51.7	70.6	89.6	**94.7**	83.6
[21]	26.1	54.1	81.8	91.0	74.6
[14]	28.6	51.2	80.0	92.0	74.4
[89]	40.2	67.8	88.7	94.3	81.0
Ours (SAMD)	**54.3**	**72.2**	**90.1**	94.6	**84.4**

Table 3.1: **Performance comparison on Audi dataset with Nvidia PilotNet [58].** All the methods are trained on RGB+segmentation, and tested on RGB only. Our method outperforms others by up to **10%** improvement in accuracy.

Sec. 3.4.2, we find the "supermodel condition", which helps AML to perform better, is not fully utilized in the general KD-based architecture. We thereby introduce a new method that uses "supermodel condition" that allows the teacher network to be aware of the student's status and leads to a better distillation.

We update the teacher-student in an online-like paradigm. See framework illustration in Fig. 4.1. The training paradigm contains t rounds. In each round, we first *reset* the teacher with the student, then train the teacher independently while training the student with both the general label loss and knowledge distillation loss for k epochs. k should not be too large to avoid the teacher being far away from the student. The training process stops when the student converges between different rounds or until finishing t rounds. See loss function, "reset" definition, and algorithm in Appendix 3.8.7.

To apply our training paradigm with *reset* operation, the framework should meet the *supermodel* condition (Sec. 3.4.2), i.e., *the teacher network should be a* supermodel *of the student network*. This condition is what differentiates our learning framework from other existing methods.

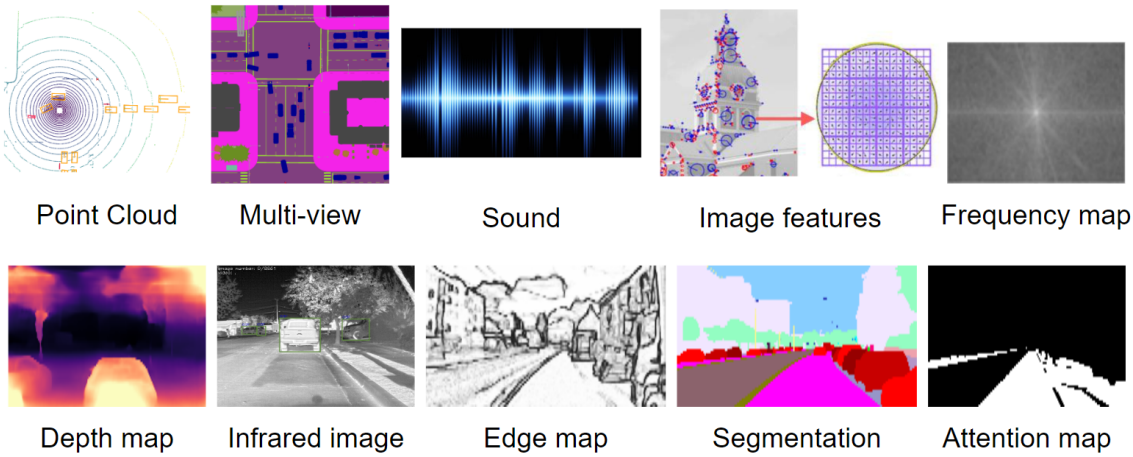


Figure 3.4: Different types of auxiliary modalities used in studies.

3.5.3 Experiments

In this section, we conduct experiments for autonomous steering task (Appendix 3.8.1) and 5 other tasks (Appendix 3.8.8). See details on the experiment settings in Appendix 3.8.8. We also use different types of data modalities in our experiments, as shown in Fig. 3.4.

Comparison with other AML methods. We compare our SAMD with other AML methods, Hoffman et al. [17], Garcia et al. [21], Xiao et al. [14], and DMCL [89], using Audi dataset [59] and Nvidia PilotNet [58]. For Xiao et al. [14], we adopt the single-sensor version and make it suitable for the Audi dataset by removing the high-level route navigation command and measurement, and using Tao et al. [105] as the segmentation generator. In Table 3.1, ours outperforms others by up to **10%**.

Effectiveness when combining with different knowledge distillation methods. Since our training paradigm can be applied to existing knowledge distillation methods, we do experiments by combining ours with kd [94], hint [97], similar-

	Mean Accuracy (%)		
Method	w/o ours	with ours	**Improvement**
kd [94]	71.5	83.4	**11.9**
hint [97]	67.6	83.2	**15.6**
similarity [106]	75.6	83.9	**8.3**
correlation [107]	77.0	74.3	-2.7
rkd [108]	75.6	84.4	**8.8**
pkt [109]	75.7	76.4	**0.7**
vid [110]	83.4	83.2	-0.2
abound [111]	74.3	72.0	-2.3
factor [112]	76.9	83.4	**6.5**
fsp [113]	72.0	70.1	-1.9

Table 3.2: **Performance comparison *with* vs. *without* our training paradigm (containing *reset* operation)**. By applying our training paradigm on other knowledge distillation methods, we can achieve better performance in most cases (up to **+15.6%**) in either fully paired or merely a small amount of additional modality data.

ity [106], correlation [107], rkd [108], pkt [109], abound [111], factor [112], fsp [113], using Audi dataset [59] and ResNet [71]. From Table 3.2, our method achieves up to **15.6%** improvement in both settings, showing the effectiveness of our training paradigm (with *reset* operation). See Appendix 3.8.8.

Comparison on different datasets and modalities. We also perform comparison with other knowledge distillation methods on different datasets (Audi [59], Honda [60], and SullyChen [24]) and different modalities (RGB, segmentation, depth map, and edge map). Specifically, Audi dataset contains ground truth segmentation, and other segmentation is generated by Tao et al. [105], while the depth map is generated by [114] and the edge map is generated by DexiNet [115]. In Table 3.3, Our method outperforms others in nearly all cases by up to **+11%** accuracy improvement. See more details in Appendix 3.8.8.

Dataset	Train Mod	Test Mod	Method	**mAcc**
Audi	RSDE	RSDE	Teacher	83.7
Audi	RSDE	RGB	Best Others	72.9
	RSDE	RGB	Ours	**74.3**
SullyChen	RDE	RDE	Teacher	81.0
SullyChen	RDE	RGB	Best Others	88.9
	RDE	RGB	Ours	**89.7**
Honda	RSDE	RSDE	Teacher	79.8
Honda	RSDE	RGB	Best Others	77.4
	RSDE	RGB	Ours	**78.1**

Table 3.3: **Comparison on different datasets and different modalities**. "RSDE" refers to RGB + segmentation + depth map + edge map, and "RDE" for RGB + depth map + edge map. Our method outperforms others on different datasets and different additional modalities by up to **+11%** accuracy improvement. "Best others" stands for the best performance among 4 methods in Table 3.1.

Comparison on other tasks and modalities. We perform comparison on multi-feature handwritten classification task [116]. We regard the six feature sets as six modalities, and treat each of them as a target modality in each experiment. Our method outperforms others with **5.1%** on average. We also conducted experiments on another end-to-end autonomous driving task, "way-point prediction" task [117]. We use RGB image as main modality, and point cloud as auxiliary modality, and achieve **19%** improvement on average route completion, compared to RGB image baseline. In the materials classification task [118], we use RGB image as main modality, while using sound wave as auxiliary modality, achieving 6.4% performance gain. For the bird-eye-view segmentation task [119], point-cloud from multiple vehicles are used during training, and point cloud from only one vehicle is used during test. We get 0.78% accuracy improvement. See Table 3.4 for a simplified comparison,

Task	Train Mod	Test Mod	Ours	Best Others
Handwritten Clas [116]	Multi-features	Single Feature	**70.3**	65.2
Waypoint Pred [117]	Image Point Cloud	Image	**79.5**	71.4
Materials Clas [118]	Image Audio	Image	**83.2**	76.8
Bird-eye-view Seg [119]	Multi-view Point Cloud	Single-view Point Cloud	**45.30**	44.91

Table 3.4: **Performance comparison on different tasks with different auxiliary modalities.** Our method outperforms other methods on all tasks. See details in Appendix 3.8.8.

and more details in Appendix 3.8.8.

Relation of Channel-level Importance and AML Performance. To show the channel-level attention for different auxiliary modalities is positively correlated to the final performance of AML with different auxiliary modalities, we conduct experiments on three tasks with the same setting stated in Sec. 3.4.1.3, then use the same three auxiliary modalities and an additional random noise modality (whose importance should be the lowest). As shown in Table 3.11, in Task 1, the importance order from the channel-level attention is attention image > depth map > frequency image, the performance order from AML is exactly the same. The same phenomenon can be observed in Task 2 and 3. This confirms that we only need to perform one-time training to select the best modality for a given task. See Appendix 3.8.8.

3.6 Conclusion

This chapter introduces '*Auxiliary Modality Learning (AML)*'. We first formalize the concept of AML in terms of types of auxiliary modality and architectures for AML. We analyze how types of auxiliary modality and architectures can affect AML performance *on a single task* and , across tasks: best architecture is consistent within a task or across tasks, while best auxiliary modality is consistent within one task but *not consistent across tasks*. We also analyze the effectiveness of AML in optimization and data perspectives to provide theory support for AML. Given these findings, we propose a novel method, SAMD, to first determine the best auxiliary modality, and then do a special auxiliary modality distillation to enable the teacher network to be aware of the student's status, leading to a better distillation that achieves the SOTA performance.

Limitations and Future Work: In modality distillation, we reasonably assume that the teacher network is a *supermodel* of the student's, as this task focuses on the reduction of modality, instead of model size, like general knowledge distillation. A possible future direction for AML is to further examine the impact of auxiliary modality data size, e.g., can we use only a small amount of auxiliary modality data to achieve better performance? What if data is not paired with the main modality? Are there better architectures? Architectures that can take unpaired input data instead of paired data would be a future direction.

3.7 Acknowledgment

This research is partially supported in part by ARO DURIP Grant, ARL Cooperate Agreement, Barry Mersky and Capital One Endowed Professorships.

3.8 Appendix

3.8.1 Details on Experimental Settings

Single task. We use autonomous driving task since there are datasets for this task that contain all types of auxiliary modality in Sec. 3.4.1.1. Specifically, the input is one RGB image and the output is one float value which represents the steering angle. Classical computer vision tasks, like object classification or detection, mostly do not use datasets with low-level sensing data other than RGB image (like depth map is not available in ImageNet or COCO). We use Audi dataset [59] and Honda dataset [60] in this experiment. Also, we use depth map (Type 1), frequency image (Type 2), and attention image (Type 3) as Auxiliary modalities. We generate depth map with [114], frequency image with standard 2D fast Fourier transform [120], and attention image with segmentation map provided by Audi dataset. We implement all four types of auxiliary modality learning architectures introduced in Sec. 3.4.1.2, and choose the Nvidia PilotNet [58] and ResNet [71] as the main backbones. Mean accuracy defined in [121] is used as the evaluation metric.

Multiple tasks. We use Audi dataset [59] for end-to-end steering task, COCO dataset [122] for real-world classification task, and a customized dataset

for customized classification task. We use semantic segmentation label contained in Audi and COCO to generate related attention images. We use blur-level estimation task as the customized task, following [121] to add blur perturbation onto the Audi dataset, and use the level ID as the ground truth, see Fig. 3.6. Also, we use attention image and frequency image as auxiliary modalities, and implement all four types of auxiliary modality learning architectures introduced in Sec. 3.4.1.2. We choose Nvidia PilotNet [58] for steering task, ResNet [71] for the classification task, and modified PilotNet for the customized classification task (change the header of the network to general classification header). We use mean accuracy [121] for steering task, accuracy for real-world classification and customized classification.

3.8.2 Experiment Results for Auxiliary Modality Types and Architectures

We show experimental results for auxiliary modality in Table 3.5 and architectures in Table 3.6. See analysis in Sec. 5.4.

3.8.3 Supermodel Example

We first introduce the "supermodel" definition:

Definition 3.8.1. *Given a model $M^{(A)}_{\theta_A}(I_A)$ (weights θ_A and input I_A), and a model $M^{(B)}_{\theta_B}(I_B)$ (weights θ_B and input I_B), if for any θ_A, there is a θ_B, such that $M^{(A)}_{\theta_A}(I_A) = M^{(B)}_{\theta_B}(I_B)$ for any arbitrary valid input data I_A and its superset I_B. We call model M_B as a* "supermodel" *of M_A.*

		Audi [59]			Honda [60]		
		Attention	Frequency	Depth	Attention	Frequency	Depth
PilotNet [58]	Archi Type A	66.3	64.9	65.8	74.5	72.9	73.4
	Archi Type B	71.6	66.5	68.4	75.9	73.2	75.1
	Archi Type C	70.1	65.7	68.8	74.3	73.7	74.2
	Archi Type D	73.4	67.9	70.8	77.4	74.8	76.7
ResNet [71]	Archi Type A	78.5	77.9	78.2	82.1	81.1	81.9
	Archi Type B	80.5	79.1	80.1	84.7	82.4	83.9
	Archi Type C	79.6	78.5	79.3	83.6	82.1	83.1
	Archi Type D	82.4	79	81.8	85.2	83	84.3

Table 3.5: Performance improvement comparison (Mean Accuracy %) with different auxiliary modalities, architectures, backbones and datasets. The relative effectiveness for different architectures is consistent under different datasets, backbones, and auxiliary modalities within one task. Similarly, The relative effectiveness for different auxiliary modalities is consistent under different datasets, backbones, and architectures within one task.

	task 1		task 2		task 3	
	Attention	Frequency	Attention	Frequency	Attention	Frequency
Archi Type A	66.3	64.9	70.1	69.3	64.3	65.2
Archi Type B	71.6	66.5	82.1	73.6	68.4	72.5
Archi Type C	70.1	65.7	80.7	71.1	65.3	70.8
Archi Type D	73.4	67.9	84.3	75.6	70.3	74.9

Table 3.6: Performance comparison (Mean Accuracy %) across tasks. The effectiveness order of different architectures is consistent across tasks, but not for auxiliary modalities.

We show a simple example of supermodel in Fig. 4.3. Net_1 contains two blocks f_1 and f_2. Net_2 contains the same block f_1 and f_2, and another block h. If there is a set of specific weights θ_0 for h that can meet $h_{\theta_0}(x) = x$ for any valid x, then Net_2 is a supermodel of Net_1, according to Definition. 4.6.1. In this case, for any specific weights of Net_1, we can always construct a set of weights for Net_2 that has exactly the same performance of Net_1, which means the optimal solution for training Net_2 will be no worse than Net_1. Furthermore, if these two models are trained in parallel, the supermodel can be "repositioned" to the same status of the base model at any

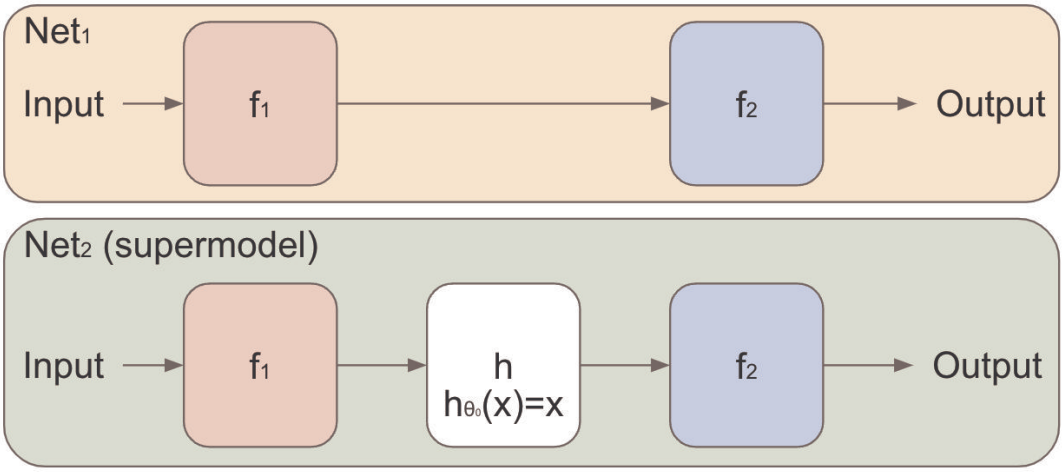


Figure 3.5: A simple example of supermodel. Net_1 contains two blocks f_1 and f_2. Net_2 contains the same block f_1 and f_2, and another block h which is possible to be set as an identical function.

time by the construction method above. This property can be used in knowledge distillation to let the teacher get back to the student's position and help find a better way at any time the student is stuck. Another example is for the same architecture with different numbers of layers, e.g., ResNet152 is a supermodel of ResNet50.

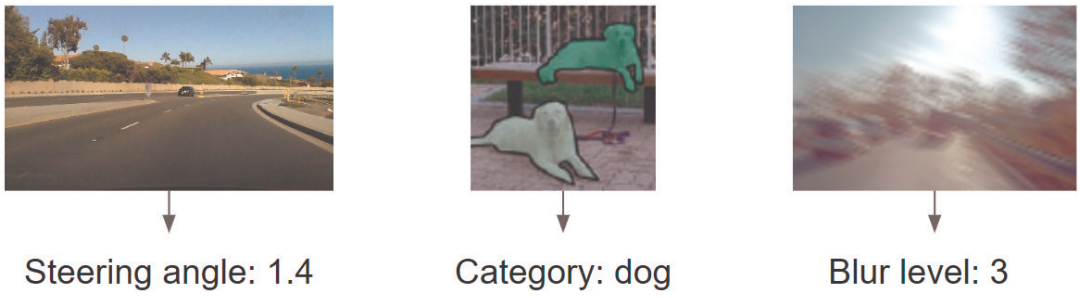


Figure 3.6: Tasks for our experiments. LEFT: end-to-end steering task, input image, output steering angle. MIDDLE: classification task, input image, output object category. RIGHT: classification task, input image, output blur level.

3.8.4 Prove of Lemma. 4.6.1

Lemma 3.8.1. *Given a model $\mathcal{M}$ and its supermodel $\mathcal{M}^{(s)}$, the optimal training loss of $\mathcal{M}^{(s)}$ (which is $\arg\min_{\theta^{(s)}} L(\mathcal{M}^{(s)}_{\theta^{(s)}}(I^{(s)}), GT)$) is less than or equal to the optimal training loss of $\mathcal{M}$ (which is $\arg\min_{\theta} L(M_\theta(I), GT)$). where L is the loss function and GT is the ground truth.*

Prove: Let $\theta^* = \arg\min_\theta L(\mathcal{M}_\theta(I), GT)$ represent the weights that lead to the best training performance for model $\mathcal{M}$, then according to the definition of supermodel, there is a $\theta^{(s)*}$ that meet $\mathcal{M}_{\theta^*}(I) = \mathcal{M}^{(s)}_{\theta^{(s)*}}(I^{(s)})$, equivalent to $L(\mathcal{M}_{\theta^*}(I), GT) = L(\mathcal{M}^{(s)}_{\theta^{(s)*}}(I^{(s)}), GT)$. That is, there's at least one solution for training $\mathcal{M}^{(s)}$ can get the same performance as training $\mathcal{M}$. Furthermore, if θ^* is the optimal solution that achieves the minimal training loss of $\mathcal{M}^{(s)}$, then the equal condition in Lemma 4.6.1 holds, if not, the less condition holds.

Notice those discussions are all on the training space, and we assume that better training performance will lead to better test performance in general. Otherwise, given the test set is unknown during training, model A is guaranteed no worse than B in test *if and only if* model A is no worse than B for every possible data points in test domain (or there will be at least one test set that contains data points that model A is worse than B), upon which no existing work can provide any theoretical guarantee.

3.8.5 More Explanation on Why AML Can Work

In Fig. 3.7, the main modality data has few examples in the hard and challenging case area, which leads to a wrong decision boundary. This is common in real-world datasets, e.g., autonomous driving datasets usually have fewer datasets for night-time driving, and even fewer on accidents. After adding the auxiliary modality that provides more information in the hard case area, it would be much easier to learn a correct decision boundary, then use this information to guide the training process with the main modality. For example, the infra-red image or depth map contains more information than RGB image when captured at night. This explains why the low-level sensing data (Type 1 in Sec. 3.4.1.1) can help AML.

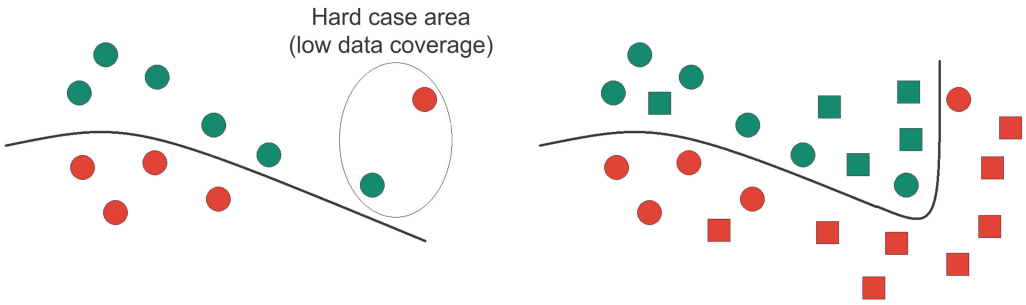


Figure 3.7: Auxiliary modality helps construct the decision boundary around the difficult cases (e.g. lack of data coverage). Circles are main modality data, and squares are auxiliary modality data.

3.8.6 Modality Choice

We show a modified network to extract channel-level importance and estimate modality effectiveness with SE block in Fig. 3.8. Suppose we already have a network f that can take the main modality I_m as input and perform prediction for a given

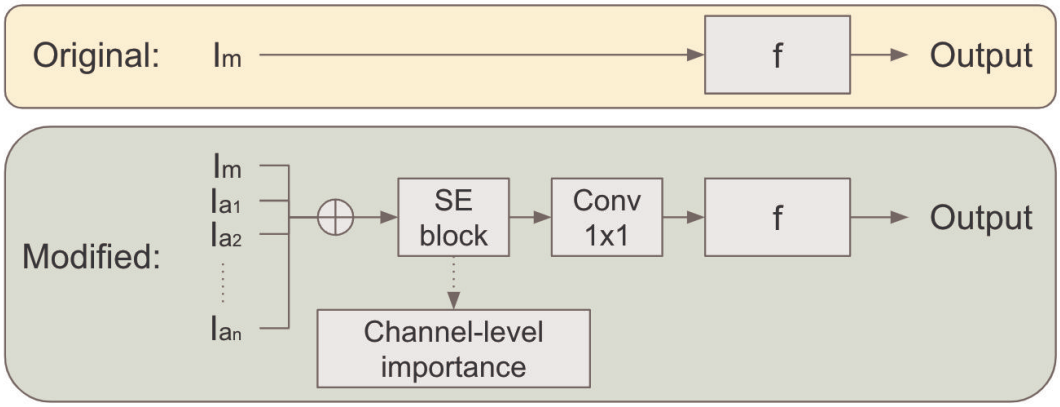


Figure 3.8: Modified network to extract channel-level importance and estimate modality effectiveness with SE block. [86]

task. Now we have n types of auxiliary modalities that potentially can help. We first pack the different modality data in the channel level, and feed them into the Squeeze-and-Excitation (SE) block [86], followed by a 1x1 convolutional layer to ensure the channel number is the same as the main modality I_m, so that the original network f can take it as input and perform prediction.

In practice, when we start to solve an AML task, we may have multiple auxiliary modality available, but collecting a full dataset for all of them may be time-consuming. We can first collect a small set of data with all modalities, and use our method to decide which or which sets of auxiliary modality is needed. After that, we can collect all the useful modality data on a larger scale, try different backbones, tune hyper-parameters, etc. Finding (5) in Sec. 3.4.1.3 motivates the need to select the best auxiliary modality at the beginning of a task, but no need to re-select even when using another architecture type, backbone, or dataset. For estimating the upper-bound, we need a full set of all modalities. The model used in this step is the

"supermodel" of the teacher model in the next step, and thus it can help estimate the upper-bound performance, given Lemma 4.6.1.

See more descriptions in Sec. 3.5.1.

3.8.7 AML in SAMD

Formally, given a task, we denote a learner composed of a feature network F and a predictor of fully-connected layers D. We design a student that takes I_M as input, and update via iterations of mini-batches,

$$\theta_{stu} \leftarrow \theta_{stu} - \eta \nabla L^M \tag{3.1}$$

where θ_{stu} is the parameter of the student network, L^M is the loss function, and η is the learning rate. Meanwhile, we design a teacher that takes $\{I_M, I_A\}$ as input, and update via an independent feature network F_{tea} (F_1, F_2, F_3 in Fig. 4.1) and a predictor D that share weights with that of the student network. The teacher network is updated via

$$\theta_{tea} \leftarrow \theta_{tea} - \eta \nabla L^A \left(D(F_{tea}(\{I_M, I_A\})), GT\right). \tag{3.2}$$

The teacher and student learn different representations related to the same task by being exposed to different modalities. The teacher has access to the auxiliary modality I_A, the knowledge of the teacher is distilled to assist the student through a consistency loss L_{con} that measures the pairwise distance between $F_{stu}(I_M)$ and

$F_{tea}(I_M, I_A)$ as part of the student's objective L^M, specifically,

$$L^M = L_{sup}\left(D(F_{stu}(I_M)), GT\right) + \beta L_{con}\left(F_{stu}(I_M), F_{tea}(\{I_M, I_A\})\right) \tag{3.3}$$

where L_{sup} is a term that supervises the learning on the main modality.

Definition 3.8.2. *Given a model $M^{(A)}_{\theta_A}(I_A)$ (weights θ_A and input I_A), and its supermodel $M^{(B)}_{\theta_B}(I_B)$ (weights θ_B and input I_B), we define "reset B with A" to be the process of constructing a new θ_B that meet $M^{(A)}_{\theta_A}(I_A) = M^{(B)}_{\theta_B}(I_B)$ for given θ_A and any arbitrary valid input data I_A and its superset I_B.*

A simple example is, suppose B is a supermodel of A (e.g., $B = A + A'$), reset B with A is constructing $\theta_B = [\theta_A, 0]$, where θ_A is the weights of A and 0 is the weights of A'. In Fig. 4.1, the teacher network is a supermodel of the student network, because for any weights of student network, we can construct a teacher network that meet $D(F_{tea}(\{I_M, I_A\})) = D(F_{stu}(\{I_M\}))$ by resetting the F_1 weights with F_4 weights, F_2 weights with F_5 weights, and set F_3 weights to 0. Indeed the reset operation in our method requires that the teacher model is a supermodel of the student model.

As shown in Algorithm 3, the training paradigm contains t rounds. In each round, we first *reset* the teacher with the student, then train the teacher independently while training student with both the general label loss and knowledge distillation loss for k epochs. k should not be too large to avoid the teacher being

far away from the student. The training process stops when the student converges between different rounds or until finishing t rounds.

Algorithm 2: SAMD Training Paradigm

Input: Training data from main modality I_M, training data from auxiliary modality I_A (chosen by method in Sec. 3.5.1)
Output: student network weights θ_{stu}
Initialisation:
Training Round number t, epoch number in each round k, loss correlation β, network weights θ_{stu} and θ_{tea}.
for $r = 1$ to t **do**
  Reset teacher weights with student weights
  for $e = 1$ to k **do**
    Feed I_M and I_A into teacher, update teacher weights θ_{tea} with Eq. 4.2
  end for
  for $e = 1$ to k **do**
    Feed I_M and I_A into teacher, and feed I_M into student, update student weights θ_{stu} with Eq. 4.1 and loss 4.3
  end for
end for=0

3.8.8 Additional SAMD Results

Setting. All experiments are conducted using one Intel(R) Xeon(TM) W-2123 CPU, two Nvidia GTX 1080 GPUs, and 32G RAM. We use the SGD optimizer with learning rate 0.001 and batch size 128 for training. The number of epochs is 2,000. The loss correlation β is set with different values for different knowledge distillation methods following [123]. We pick epoch number in each round $k = 5$ from ablation study of $k = 1, 2, 5, 20$. We set the round number $n = 400$ for Audi dataset and $n = 40$ for Honda dataset. In the experiments, each training process is finished within 24 hours. The main task is the steering task introduced in the single

task setting in Appendix 3.8.1.

Comparison on other tasks. To show the generalizability of our method, we perform comparison on multi-feature handwritten classification task [116] in Table 4.4. The dataset [124] consists of six features of handwritten numerals ('0'–'9') with 2,000 samples in total. We regard the six feature sets as six modalities, and treat each of them as target modality in each experiment. Our method outperforms others by 5.1% higher accuracy on average.

	Accuracy (%) on different modalities (ID:1∼6)						
Method	1	2	3	4	5	6	mean
Best Others	84.92	62.98	68.75	61.10	70.35	43.17	65.2
Ours	**89.40**	**65.20**	**72.80**	**69.50**	**73.15**	**51.75**	**70.3**

Table 3.7: **Performance comparison on handwritten classification task.** Our method outperforms other KD methods listed in Table 3.1 by 5.1% higher accuracy *on average*.

We also conducted experiments on another end-to-end autonomous driving task, "way-point prediction" task. Following the setting of [117], we consider the task of navigation along a set of predefined routes in different areas, such as motorways, urban regions, and residential districts. A sequence of sparse goal locations in GPS coordinates, provided by a global planner and the related discrete navigational commands (e.g. "follow lane", "turn left/right", and "change lane"), constitute the routes. Only the sparse GPS locations are used in our method. Each route consists of several scenarios, which are initialized at predefined locations and test the agent's ability to handle various adversarial situations, such as obstacle avoidance, unprotected turns at intersections, vehicles running red lights, and pedestrians

Model	DS↑	RC↑	IP↓	CP↓	CV↓	CL↓	RLI↓	SSI↓
RGB	21.0	60.5	0.49	0.01	0.15	0.08	**0.14**	0.04
RGB+PC	11.2	52.9	0.37	0.02	0.22	**0.01**	0.38	**0.02**
Ours(new)	**22.1**	**79.5**	0.37	0.01	**0.07**	0.04	0.26	0.04

Table 3.8: **Performance comparison on long-route waypoints prediction** between base (train and test on RGB), multi-modality (train and test on RGB + point cloud), and ours (train on RBG + point cloud, test using *only RGB*). DS: Avg. driving score, RC: Avg. route completion, IP: Avg. infraction penalty, CP: Collisions with pedestrians, CV: Collisions with vehicles, CL: Collisions with layout, RLI: Red lights infractions, SSI: Stop sign infractions.

emerging from occluded regions crossing the road at random locations. The agent needs to complete the route within a certain amount of time, while following traffic regulations and dealing with large numbers of dynamic agents. For dataset, we use the CARLA [125] simulator for training and testing, specifically CARLA 0.9.10 which includes 8 publicly available towns. We use 7 towns for training and hold out Town05 for evaluation, as in [117]. We use both RGB and LiDAR for training in AML, but *only RGB* data for testing. The results are shown in Table 3.8. Our method benefits from the auxillary LiDAR modality in training using AML, with only RGB data during query. This set of experimental results demonstrates the effectiveness of AML.

In addition, we apply our method on audio modality based on an audio-visual depth and material estimation work [118]. We use RGB image as the main modality, and audio wave as the auxiliary modality. The task is material and depth classification. We use the same dataset in the original audio-visual work, which contains about 16,000 pairs of RGB image and audio wave. Since there's no open-source code, we reimplement the original work, then apply our method to it. Our method

	Accuracy (%) on various angle threshold τ (degree)					
Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 75$	**mAcc**
Seg GT	50.6	70.9	85.4	96.1	99.2	80.44
Seg Infer	48.3	69.5	85.3	95.7	98.6	79.48

Table 3.9: **Performance comparison between ground truth and generated segmentation**. The results show that the inferred segmentation can do nearly as well as ground truth segmentation, when serving as auxiliary modality (within 1% of difference). Therefore, we can use pre-trained models to generate auxiliary modality conveniently.

Method	Vehicle	Sidewalk	Terrain	Road	Building	Pedestrian	Vegetation	mIoU
Lower-bound	45.93	42.39	47.03	65.76	25.38	20.59	35.83	40.42
Co-lower-bound	47.67	48.79	50.92	70	25.26	10.78	39.46	41.84
When2com [127]	48.43	33.06	36.89	57.74	29.2	20.37	39.17	37.84
Who2com [128]	48.4	32.76	36.04	57.51	29.17	20.36	39.08	37.62
DiscoNet [126]	**56.66**	46.98	**50.22**	**68.62**	27.36	22.02	42.5	44.91
Ours	56.52	**47.43**	49.72	67.72	**30.59**	**22.23**	**42.86**	**45.30**
Upper-bound	64.09	41.34	48.2	67.05	29.07	31.54	45.04	46.62

Table 3.10: **Performance comparison on bird-eye-view segmentation task.** Our method achieves the best performance compared to three other methods, with only 1.32% performance difference from the upper-bound. We follow the same setting of [126] for the lower-bound, co-lower-bound and upper-bound.

outperforms other KD methods listed in Table 3.1 by 6.4%.

Finally, we apply our method on a bird-eye-view segmentation task [119]. During training, a mixed point cloud from multiple viewpoints is used as input, while a point cloud from one viewpoint is used during test. We use the same virtual autonomous driving dataset [119], which contains 48,000 datapoints for training, 6,000 datapoints for test, and 6,000 datapoints. We apply our method based on the DiscoNet [126]. In Table 3.10, we show our method achieves the best performance compared to other methods.

Comparison on different datasets and modalities. We also perform comparison with other knowledge distillation methods on different datasets (Audi [59],

	Channel-level Importance				AML Performance			
	Attention	Frequency	Depth	Noise	Attention	Frequency	Depth	Noise
Task 1	0.32	0.08	0.12	6.9e-6	73.4	67.9	70.8	65.2
Task 2	0.65	0.12	-	2.7e-6	84.3	75.6	-	70.1
Task 3	0.09	0.26	-	3.2e-6	70.3	74.9	-	60.8

Table 3.11: **Relation between relative orders of channel-level importance and AML performance for different auxiliary modalities.** The relative modality orders are consistent between channel-level importance and AML performance within each task, therefore we can use channel-level importance to choose the best auxiliary modality before AML.

Honda [60], and SullyChen [24]) and different modalities (RGB, segmentation, depth map, and edge map). Specifically, Audi dataset contains ground truth segmentation, and other segmentation is generated by Tao et al. [105], while the depth map is generated by [114] and the edge map is generated by DexiNet [115]. In Table 4.5, our method outperform others in practically all cases by up to **+11%** accuracy improvement.

Effectiveness when combining with different knowledge distillation methods. Since our training paradigm can be applied on existing knowledge distillation methods, we do experiments by combining ours with kd [94], hint [97], similarity [106], correlation [107], rkd [108], pkt [109], abound [111], factor [112], fsp [113]. From Table. 4.2, our method achieves up to 15.6% improvement in both settings, showing the effectiveness of our training paradigm (containing *reset* operation).

Relation of Channel-level Importance and AML Performance. To show the channel-level attention for different auxiliary modalities is positively correlated to the final performance of AML with different auxiliary modalities, we conduct experiments on three tasks with the same setting stated in Sec. 3.4.1.3, then use the

same three auxiliary modalities and an additional random noise modality (whose importance should be the lowest). We use knowledge distillation based architecture (Type D), since it's consistently better than other architectures (see Sec. 3.4.1.3).

As shown in Table 3.11, in Task 1, the importance order from the channel-level attention is attention image > depth map > frequency image, and the performance order from AML is also attention image > depth map > frequency image. The same phenomenon can be observed in Task 2 and 3. This shows we only need to perform one-time training to select the best modality for a given task.

Comparison of ground truth and generated auxiliary modality. We conduct experiment with ground truth segmentation and generated segmentation [105] to see how much it will influence the performance. The model used to generate segmentation for Audi dataset [59] is trained on Cityscapes dataset [129]. Table 3.9 shows that the generated segmentation can do nearly as well as ground truth segmentation, when serving as auxiliary modality (i.e. within 1% of difference), thus we can use pre-trained models to generate auxiliary modality conveniently.

3.8.9 Tasks, Datasets, Backbones

Tasks. We use *autonomous driving tasks and 5 additional tasks in other domains*. These include: object classification in the multi-task experiment (Sec. 3.4.1.3), handwritten classification, waypoint prediction, materials classification, and bird-eye-view segmentation experiments (in Table 3.4 from Sec. 3.5.3, and Appendix 3.8.8).

				Accuracy (%) on different angle threshold τ (degree)						
Dataset	Train Mod	Test Mod	Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**mAcc**
Audi	RGB+seg	RGB+seg	Teacher	42.7	68.0	88.0	94.4	96.6	98.6	81.4
Audi	RGB+seg	RGB	best others	30.3	51.0	78.2	88.4	94.4	98.2	73.4
	RGB+seg	RGB	ours	**52.6**	**72.7**	**91.3**	**95.0**	**97.0**	**98.3**	**84.5**
Audi	RSDE	RSDE	Teacher	49.9	72.1	89.5	94.9	97.1	98.6	83.7
Audi	RSDE	RGB	best others	27.7	47.8	77.4	90.8	95.6	98.3	72.9
	RSDE	RGB	ours	**30.2**	**50.3**	**79.7**	**91.0**	**96.2**	**98.6**	**74.3**
SullyChen	RDE	RDE	Teacher	41.1	63.7	88.6	95.9	97.9	99.1	81.0
SullyChen	RDE	RGB	best others	59.5	82.1	93.9	98.2	99.5	100.0	88.9
	RDE	RGB	ours	**63.4**	**83.0**	**94.3**	98.2	99.5	100.0	**89.7**
Honda	RSDE	RSDE	Teacher	41.3	61.1	83.9	94.0	98.3	99.9	79.8
Honda	RSDE	RGB	best others	**38.9**	**57.7**	79.7	91.7	97.5	99.3	77.4
	RSDE	RGB	ours	37.9	**57.7**	**81.7**	**93.5**	**98.2**	**99.6**	**78.1**

Table 3.12: **Comparison on different datasets and different modalities**. "RSDE" refers to results from RGB + segmentation + depth map + edge map, and "RDE" for RGB + depth map + edge map. Our method outperforms others on different datasets and different additional modalities by up to **+11%** accuracy improvement.

Datasets. We use **10 datasets** in total. They are: Honda, Audi, COCO, a customized dataset (Sec. 3.4.1.3), SullyChen Driving data, CityScapes, 4 datasets for handwritten classification (described in Appendix 3.8.1), waypoint prediction, materials classification, and bird-eye-view segmentation, used in Sec. 3.5.3 and in Appendix 3.8.8.

Backbones. We conduct experiments and analysis on **8 backbones** in total. They are: (1) PiloNet and (2) ResNet for steering and object classification (Sec. 3.4.1.3), (3) TMC for handwritten classification, (4) Multi-Modal Fusion Transformer for waypoint prediction, (5) EchoCNN-AV for materials classification, (6-8) When2com, Who2com, and DiscoNet for bird-eye-view segmentation (Sec. 3.5.3).

	Accuracy on different threshold τ (%)							
Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**Mean**	**Improvement**
Train Vanilla								
Teacher (img+seg)	42.7	68.0	88.0	94.4	96.6	98.6	81.4	
Student (img)	27.3	49.0	77.4	90.2	95.4	98.1	72.9	
Existing Distillation Methods								
kd [94]	28.4	47.7	73.2	87.2	94.3	98.4	71.5	
hint [97]	31.7	50.2	69.5	77.0	83.7	93.8	67.6	
similarity [106]	33.0	55.9	80.8	90.5	95.1	98.3	75.6	
correlation [107]	36.2	59.1	81.5	91.7	95.3	98.2	77.0	
rkd [108]	32.9	53.6	80.3	91.8	96.2	98.5	75.6	
pkt [109]	34.2	55.4	80.8	90.4	94.9	98.5	75.7	
vid [110]	49.7	71.2	89.9	94.8	96.7	98.3	83.4	
abound [111]	32.8	53.9	77.8	88.9	94.6	98.0	74.3	
factor [112]	36.8	59.2	82.0	90.6	94.7	97.9	76.9	
fsp [113]	30.8	51.6	74.9	85.8	91.6	97.4	72.0	
Existing Distillation Methods with Our Training Paradigm								
kd [94]	49.7	71.2	89.9	94.8	96.7	98.3	83.4	**11.9**
hint [97]	48.6	71.0	90.1	94.8	96.7	98.3	83.2	**15.6**
similarity [106]	52.1	71.8	90.0	94.8	96.6	98.3	83.9	**8.3**
correlation [107]	31.8	52.7	78.1	89.7	95.2	98.3	74.3	-2.7
rkd [108]	54.3	72.2	90.1	94.7	96.6	98.3	84.4	**8.8**
pkt [109]	34.5	56.9	82.9	90.3	95.5	98.4	76.4	**0.7**
vid [110]	48.6	71.0	90.1	94.8	96.7	98.3	83.2	-0.2
abound [111]	29.6	49.5	74.4	87.3	93.5	97.8	72.0	-2.3
factor [112]	49.7	71.2	89.9	94.8	96.7	98.3	83.4	**6.5**
fsp [113]	28.8	48.2	71.5	83.9	91.2	97.4	70.1	-1.9

Table 3.13: **Performance comparison *with* vs. *without* our training paradigm (containing *reset* operation)**. By applying our training paradigm on other knowledge distillation methods, we can achieve better performance in most cases (up to **+15.6%**) in either fully paired or merely a small amount of additional modality data.

3.8.10 Dataset Description

Honda dataset [60], or HRI Driving Dataset (HDD), is a challenging dataset to enable research on learning driver behavior in real-life environments. The dataset includes 100+ long-time driving videos with 104 hours of real human driving in the San Francisco Bay Area collected using an instrumented vehicle equipped with different sensors. We first select 30 videos that are most suitable for learning to steer task, then we extract 110,000 images from them at 1 FPS, and align them

with the steering labels.

Audi dataset [59], or Audi Autonomous Driving Dataset (A2D2), is a dataset that features 2D semantic segmentation, 3D point clouds, 3D bounding boxes, and vehicle bus data. It includes more than 40,000 frames with semantic segmentation image and point cloud labels, of which more than 12,000 frames also have annotations for 3D bounding boxes. In addition, the authors provide unlabelled sensor data (approx. 390,000 frames) for sequences with several loops, recorded in three cities. In our experiment, we use the "Gaimersheim" package which contains about 15,000 images with about 30 FPS. For efficiency, we adopt a similar approach as in [58] by further downsampling the dataset to 15 FPS to reduce similarities between adjacent frames, keep about 7,500 images and align them with steering labels.

SullyChen dataset [24] is designed for the steering task with the longest continuous driving image sequence without road branching. Images are sampled from videos at 30 frames per second (FPS). We downsample the dataset to 5 FPS. The resulting dataset contains ≈10,000 images.

COCO [122] is a large-scale object detection, segmentation, and captioning dataset. COCO has several features: Object segmentation, Recognition in context, Superpixel stuff segmentation, 330K images (¿200K labeled), 1.5 million object instances, 80 object categories, 91 stuff categories, 5 captions per image, 250,000 people with keypoints.

Other datasets used in Table 3.4. Handwritten classification dataset [124] consists of six features of handwritten numerals ('0'–'9') with 2,000 samples in total. In the end-to-end autonomous driving task, we use the CARLA [125] simulator for

training and testing, specifically CARLA 0.9.10 which includes 8 publicly available towns. We use 7 towns for training and hold out Town05 for evaluation, as in [117]. In the audio-visual depth and material estimation work [118], we use the same dataset in the original audio-visual work, which contains about 16,000 pairs of RGB images and audio waves. In the bird-eye-view segmentation task [119], we also use the same virtual autonomous driving dataset [119], which contains 48,000 datapoints for training, 6,000 datapoints for test, and 6,000 datapoints.

3.8.11 More Related Works

Except for the cross-modality learning and knowledge distillation works introduced in Sec. 4.2.2, there are other related works from curriculum distillation, multimodal learning and auxiliary learning.

Curriculum distillation aims to do knowledge distillation in a curriculum way. Jin et al. [99] proposes RCO that supervises the student model with some anchor points selected from the parameter space route that the teacher model passed by, while ours is using *online* distillation with start points selected from the parameter space route that the student model passed by. Xiang et al. [130] do curriculum on *instance* level with *multiple* teachers, Li et al. [131] do curriculum on *hyper-parameter* level (which is the temperature for knowledge distillation) with one teacher, while ours do curriculum on *parameter* level with one teacher.

Multimodal learning works [132] use the same types of modality during training and test, but ours focus on modality reduction. Some of them [133, 134] use matrix-based fusion, some [135] use MLP-based fusion, and some [136, 137] use attention-

based fusion.

AML improves the ability of a primary task to generalize to *unseen* data, by training on additional auxiliary tasks alongside this primary task, while ours don't have multiple tasks. For example, Liebel et al. [138] propose a method that using auxiliary task to boost the performance of the ultimately desired main tasks, Valada et al. [139] propose VLocNet, a new convolutional neural network architecture for 6-DoF global pose regression and odometry estimation from consecutive monocular images, and recently Chen et al. [140] propose to learn a joint task and data schedule for auxiliary learning, which captures the importance of different data samples in each auxiliary task to the target task.

Chapter 4: Small-shot Multi-modal Distillation for Vision-based Autonomous Steering

4.1 Introduction

The core component of self-navigation systems is *autonomous steering* that requires both correct scene understanding and rapid adaptation to the changing circumstances. Because of the variant scenarios in autonomous driving, people explore the possibility of seeking auxiliary information instead of single sensor information to improve the learning of the autonomous steering task. Previous works [14, 15, 16] have tried to exploit the depth information in addition to the RGB channels, such as Lidar. The unified learning framework that involves multiple modalities of data as input is referred as multi-modality learning. However, it is computationally very expensive. Also, the framework that requires the auxiliary sensor/data for input at test time largely restricts its application to cars with less advanced equipment. Another problem is, the amount of auxiliary information may be small in some cases, e.g., expensive expert-labeled data, or sensing data from a low-frequency sensor, which makes the network harder to learn. Therefore, our aim in this work is to design a learning framework that *utilizes a small amount of auxiliary sensor/data*

to assist the task during training, but does not require it during test/inference time.

In this chapter, we introduce a novel learning framework for autonomous steering that uses a small amount of "auxiliary modality" data to complement the learning of the main modality, i.e., distilling knowledge from a multi-modality teacher to a single-modality student with *partially available auxiliary modality*. Specifically, we propose a small-shot auxiliary modality distillation network, **AMD-S-Net** (Sec. 4.3.2), for the partially available setting, which is trained with our multi-modality training paradigm and meets a special "supermodel condition". It uses a special "reset operation" that allows a teacher to be aware of the exact student states (Sec. 4.3.3). In addition, another novelty of the AMD-S-Net framework design is the classification of the input data into two types. We design a specific framework for each type of data, according to their special properties: (1) *"consistency supervision"* for the *pairwise data* and (2) *"distribution divergence supervision"* for the *unpaired data* – to fully extract information in each data type (Sec. 4.3.2).

Furthermore, general knowledge distillation methods do not ensure that the teacher is aware of the student's states. This implies that the teacher itself may learn well, but not necessarily teach well. Consider the difference between letting a teacher teach by recording videos vs. by interacting with students. We hereby propose a multi-round online-distillation training paradigm (Sec. 4.3.4) that utilizes the "reset operation" which can ensure that the teacher is aware of the exact student states (e.g., learning process, features, loss, etc). In each round, our training paradigm will first reset the teacher's to the student's states, then let the teacher learn independently in a higher dimensional space to explore the loss landscape near

the student space, and guide the student with the local landscape information and potential direction of a better solution, leading to better student performance. This is an advantageous property of the teacher, when the student has converged to a relatively small empirical loss and is unable to further optimize in a stochastic local search.

Experiments show that our AMD-S-Net architecture outperforms other architectures by up to **12.7%**, and our training paradigm outperforms other knowledge distillation (KD) methods by up to **18.1%**. We conduct comparisons on 5 architectures, 10 KD methods, 5 backbones, 4 datasets, and 5 different auxiliary modalities to show their effectiveness. We also perform experiments on other tasks, including waypoints prediction (using RGB + point clouds) and handwritten classification (images + non-image features) to illustrate the generalizability of our method (see Sec. 4.4).

We summarize our key contributions as follows:

- We propose a novel framework that distill knowledge from multi-modality model to single-modality model in a partially available auxiliary modality setting, i.e., small-shot auxiliary modality distillation network, **AMD-S-Net**. AMD-S-Net contains a specific framework design to fully distill the information, i.e., *consistency supervision* for the pairwise data and *distribution divergence supervision* for unpaired data (Sec. 4.3.2).

- We propose a novel knowledge distillation training paradigm (Sec. 4.3.4) that enables teachers to explore and learn student's local loss landscape information

in a higher dimension, thus making it feasible to help student get out of local minimal and boost performance, based on a special "reset operation" that allows the teacher to be aware of the exact student states.

4.2 Related Work

In our paper, we mainly focus on the setting that there are auxiliary modalities during training but only main modality during test. One kind of solution is treating the multimodal network as the teacher and single modality network as the student, and use the general knowledge distillation methods (Sec. 4.2.1) to transfer the knowledge. Another kind of solution is designing architectures specifically for the modality distillation (Sec. 4.2.2). For the task, we choose end-to-end learning steering under multimodal settings in general (Sec. 4.2.3).

4.2.1 Knowledge Distillation

Knowledge distillation is the process to transfer knowledge between networks [94]. Many works have already been done in the general knowledge distillation area. Hinton et al. [94] do early research about distilling the knowledge from an ensemble of models to a single model. Then more and more works have explored the desired knowledge need to be distilled, including intermediate layers' feature [113, 141], attention map [142], paraphrased feature [112], probability distribution in the feature space [109], activation of neurons [111] and etc. Romero et al. [97] propose a method that can distill knowledge from a wide shallow network to a deep thin net-

work (FitNet). VID [110] formulates knowledge transfer as maximizing the mutual information between the teacher and the student networks. Similarity-Preserving Knowledge Distillation [106] aims to preserve the similarity matrix of input data within a mini-batch. CRD [123] encourages the teacher and student to map the same input to close representations and different inputs to distant representations. Some other works [107, 108] focuses on correlation congruence between data samples instead of instance congruence.

Our method can be combined with these methods and achieve better performance. Specifically, our method can reset the teacher to the student's states and lead the student step by step, making it possible to escape local minima and achieve better performance. This is different from the self-distillation methods (e.g., [143, 144]), where teacher and student share the same architecture, while in our setting teacher and student do not need to have the same network architecture.

4.2.2 Modality Distillation

Modality distillation mainly focuses on distilling knowledge between different modalities. Gupta et al. [87] learn the representation of one modality with a pretrained network on another modality. Hoffman et al. [17] do an early work about modality hallucination, which contains a hallucination network with RGB image as input but tries to mimic a depth network, then combines with RGB network to achieve multimodal learning. Following works [18, 21] train the hallucination network with a different process to achieve better performance. Some other

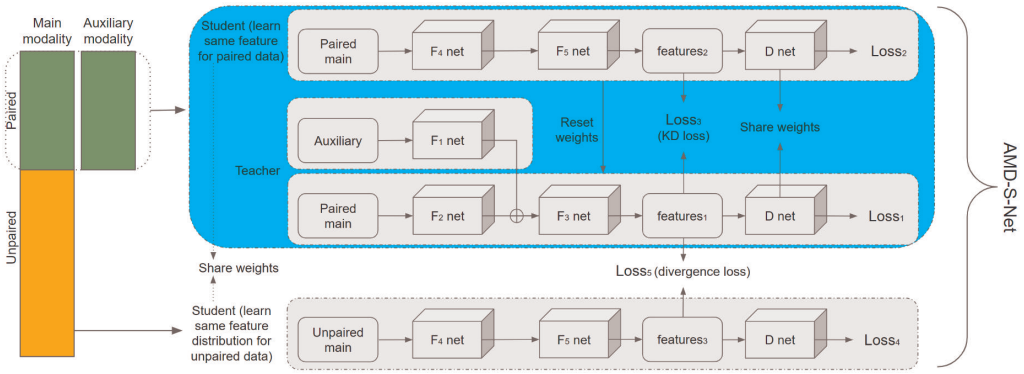


Figure 4.1: **AMD-S-Net Architecture.** The training process consists of t rounds and each round contains k epochs. At the beginning of each round, the student network will be used to initialize the teacher. In each round of the AMD-S-Net training process, there are 2 steps: (1) Calculate the teacher's loss, $Loss_1$; backpropagate and update teacher's networks F_1, F_2, F_3, D for k epochs; (2) Feed the paired main modality data to the student, calculate the student's loss, $Loss_2$, and feature loss, $Loss_3$, update student's networks F_4, F_5, D, and feed unpaired main modality data to the student, calculate student's loss, $Loss_4$, and divergence loss, $Loss_5$, update F_4, F_5, D, train for k epochs.

works [19, 20] use GAN or U-Net to generate another paired modality data with one modality. MSD [88] transfers knowledge from a teacher on multimodal tasks by learning the teacher's behavior within each modality. A latest work [89] trains the different modality data in different pipelines and distills the best modality pipeline knowledge to other modality pipelines. Other than action recognition, modality distillation has also been applied in medical image processing [90]. Existing work of unpaired modality distillation like [145, 146] only consider unpaired data and assume both modalities have enough samples, while ours consider both paired and unpaired data, and also only have small number of auxiliary modality data. Compared to these methods, our method is the first framework that uses consistency supervision for the pairwise data and distribution divergence supervision for the unpaired data (Sec. 4.3.2) and provide flexibility for real-world applications.

4.2.3 Multimodal End-to-end Steering

End-to-end steering is an essential task in end-to-end autonomous driving [58]. Multimodal end-to-end steering becomes popular, because of its naturally abundant information and the improvement of multimodal architectures.

Xiao et al. [14] analyze different architectures to fuse multiple modalities in the simulator. Yang et al. [147] make the multimodal data to be the supervision of their multimodal multitask network with only image input. Huang et al. [15] propose a multimodal method with scene understanding. Recently Maanpää et al. [16] design a specific network to fuse camera and lidar data that are suitable for adverse road and weather conditions. Except for spatial methods, Abou-Hussein et al. [148] propose an LSTM-based network to utilize multimodal Spatio-Temporal information.

Compared to these works, ours considers the multimodal end-to-end steering in a more specific setting, i.e., there is a varying amount of auxiliary modality data during training that can reduce costs compared to general multimodal methods while outperforming single-modality techniques.

4.3 Approach

In this section, we first introduce the task formalization in Sec. 4.3.1. Next, we explain our method in detail in Sec. 4.3.2 (AMD-S-Net) under different settings. We introduce a specific *reset* operation and *supermodel* condition in Sec. 4.3.3, which is used by our training paradigm in Sec. 4.3.4.

The novelty of AMD-S-Net is that it's trained with our novel training paradigm and should satisfy the *supermodel* condition to ensure their suitability for our training paradigm with *reset* operation. The framework of AMD-S-Net is also novel because of a specific two-stream framework design. This is the first work that introduces a *reset* operation and *supermodel* condition that can be utilized by the training paradigm to boost performance.

4.3.1 Auxiliary Modalities and Task Formalization

Given an arbitrary task that can be learned by observing a series of task-related data captured by different sensors, or processed using different techniques, we refer to these different but related data types as modalities $\mathbb{I} = \{I_k\}_{k=1}^{K}$, where K is the maximum number of modalities one can obtain with the existing devices, signal prepossessing methods, or expert annotation. We assume that among the K modalities, there is one *main modality* I^M that contributes the most information to the task. The modalities other than I^M are referred as *auxiliary modalities* $\mathbb{I}^A$. Note that each sample from the I^M is not necessarily more informative than each sample from an auxiliary modality. The main modality I^M is considered primary usually because it is the most available hence used-at-*inference* data type. One example of the main modality is the data captured with RGB cameras for autonomous driving tasks, which is plentiful and not expensive, but not necessarily more informative than depth cameras [14, 149].

We first consider a model with learnable parameters θ^m that prioritizes the

data from the main modality. The training data from I^M is denoted as $\mathcal{I}^M_{train} = \{i^M_n\}^{N_{train}}_{n=1}$, where N_{train} is the number of training samples from I^M. The parameter that achieves the smallest inference error ϵ^M on $\mathcal{I}^M_{test}$ is denoted as θ^{M*}. With additional data from auxiliary modalities joining the training process, a new model is learned using $\mathcal{I}^{train} = \mathcal{I}^M_{train} \cup \mathcal{I}^A$. The question is, "can we find a better model that achieves a lower inference error on $\mathcal{I}^M_{test}$". In other words, *our goal is to distill complementary information from the auxiliary modalities at training to achieve higher accuracy at test time.*

4.3.2 Small-shot Auxiliary Modality Distillation Network (AMD-S-Net)

We first consider the training samples that can find paired matches from both the auxiliary modality and main modality. Our goal is to distill the knowledge from any arbitrary paired I^A and I^M that improves the model that later inference on I^M.

Formally given a task, we denote a learner composed of feature network F and a predictor of fully-connected layers D. We design a student that takes $\mathcal{I}^M_{train}$ as input, and update via iterations of mini-batches,

$$\theta_{stu} \leftarrow \theta_{stu} - \eta \nabla \mathcal{L}^M \tag{4.1}$$

where θ_{stu} is the parameter of the student network, $\mathcal{L}^M$ is the loss function, and η is the learning rate. Meanwhile, we design a teacher that takes $\{\mathcal{I}^M_{train}, \mathcal{I}^A\}$ as

input, and update via an independent feature network F_{tea} (F_1, F_2, F_3 in Fig. 4.1) and a predictor D that share weights with that of the student network. The teacher network is updated via

$$\theta_{tea} \leftarrow \theta_{tea} - \eta \nabla \mathcal{L}^A \left(D(F_{tea}(\{i_n^M, i_n^A\})), y_n^M \right). \tag{4.2}$$

The teacher and student learn different representations related to the same task by being exposed to different modalities. The teacher has access to the auxiliary modality I^A, the knowledge of the teacher is distilled to assist the student through a consistency loss $\mathcal{L}_{con}$ that measures the pairwise distance between $F_{stu}(i_n^M)$ and $F_{tea}(i_n^M, i_n^A)$ as part of the student's objective $\mathcal{L}^M$, specifically,

$$\begin{aligned} \mathcal{L}^M = \alpha \mathcal{L}_{sup} \left(D(F_{stu}(i_n^M)), y_n^M \right) + \\ \beta \mathcal{L}_{con} \left(F_{stu}(i_n^M), F_{tea}(\{i_n^M, i_n^A\}) \right) \end{aligned} \tag{4.3}$$

where $\mathcal{L}_{sup}$ supervises the learning on the main modality.

When auxiliary data is hard to obtain, utilizing a small amount of paired auxiliary data based on the main data is an alternative. We refer to distillation under such a condition as *small-shot auxiliary modalities distillation*. Data modalities such as intermediate annotations, expert commentary for hard examples, etc. usually come in *small amounts but are exceptionally informative*, e.g. the doctor's coarse annotation of medical images for tumor segmentation, or human-in-the-loop interactive systems [150]. Except for the consistency supervision by the pairwise

feature distance, we also use a *divergence metric* to estimate the difference of the distributions for the unpaired data, such as Kullback–Leibler divergence [151]. During the training (Sec. 4.3.4), after updating the student network via loss, as defined in Eq. 4.1, for all paired data, we update the student network again with unpaired main modality data via the following loss:

$$\mathcal{L}_u^M = \gamma \mathcal{L}_{sup-u} \left(D(F_{stu}({}_{up}i_m^M)), y_n^M \right) + \lambda \mathcal{L}_{div} \left(\{F_{stu}({}_{up}i_m^M)\}, \{F_{tea}(\{{}_{p}i_n^M, i_n^A\})\} \right) \tag{4.4}$$

where $\{{}_{p}i_n^M\}$ and $\{{}_{up}i_m^M\}$ are paired and unpaired main modality data that meet $\{{}_{p}i_n^M\} \cup \{{}_{up}i_m^M\} = \mathcal{I}_{train}^M$, and $\mathcal{L}_{div}(,)$ measures the divergence between the distributions of the feature representation sets $\{F_{stu}({}_{up}i_m^M)\}, \{F_{tea}(\{{}_{p}i_n^M, i_n^A\})\}$. Other training process is shared with the paired process. See this AMD-S-Net framework illustration in Fig. 4.1 and Algorithm. 3.

One key consideration of this design is, what kind of information is important under this problem setting (input data in Fig. 4.1). One is the relation between paired main modality feature and the combination feature of paired main and auxiliary modality, which can be extracted by the paired data using consistency supervision knowledge distillation. Another one is the relation between the unpaired main modality feature and the combination feature of main and auxiliary modality. Since the auxiliary modality data is missing for the unpaired main modality data, the combination feature is actually *unknown*. Thus we use the distribution space of combination features of paired main and auxiliary modality to be an approxi-

Algorithm 3: AMD-S-Net Training Paradigm

Input: Training data from main modality ${}_{p}\mathcal{I}^{M}_{train}$ (with paired auxiliary data) and ${}_{up}\mathcal{I}^{M}_{train}$ (no paired auxiliary data), training data from auxiliary modality $\mathcal{I}^{A}$ (paired with ${}_{p}\mathcal{I}^{M}_{train}$)
Output: student network weights θ_{stu}
Initialisation:
Training Round number t, epoch number in each round k, loss correlation $\alpha, \beta, \gamma, \lambda$, network weights θ_{stu} and θ_{tea}.
for $r = 1$ to t **do**
  Reset teacher weights with student weights
  for $e = 1$ to k **do**
    Feed ${}_{p}\mathcal{I}^{M}_{train}$ and $\mathcal{I}^{A}$ into teacher, update teacher weights θ_{tea} with Eq. 4.2
  end for
  for $e = 1$ to k **do**
    Feed ${}_{p}\mathcal{I}^{M}_{train}$ and $\mathcal{I}^{A}$ into teacher, and feed ${}_{p}\mathcal{I}^{M}_{train}$ into student, update student weights θ_{stu} with Eq. 4.1 and loss 4.3
    Feed ${}_{up}\mathcal{I}^{M}_{train}$ into student, update student weights θ_{stu} with Eq. 4.1 (replace loss 4.3 with loss 4.4).
  end for
end for=0

mation of the unknown distribution space of combination features of the unpaired data. Also, since we don't have one-to-one mapping for unpaired data, we use divergence supervision on the distribution-level, instead of consistency supervision on the sample-level. To the best of our knowledge, our AMD-S-Net is the first method that uses consistency supervision for pairwise data and distribution divergence supervision for unpaired data, making this method unique and different from others.

4.3.3 Reset Operation

Lemma[section] Definition[section]

The reset operation plays an important role in our method, but a condition is needed to apply this operation. Inspired by "superset", we introduce "supermodel".

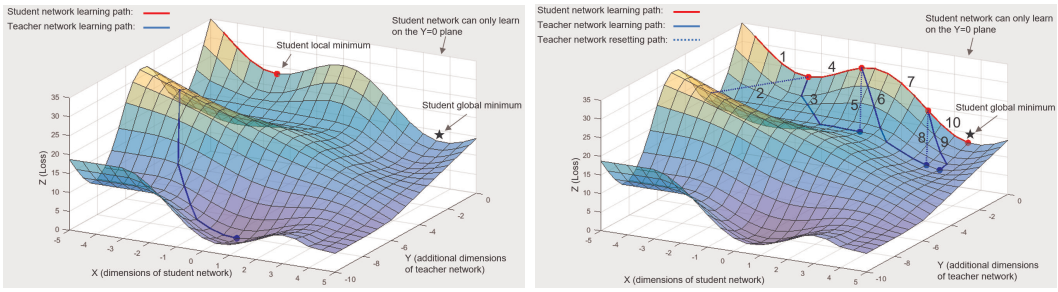


Figure 4.2: **Training Path Comparison on Loss Landscape.** Given the teacher network is a *supermodel* of the student network, the student parameter space (along X axis with Y=0) is a subspace of the teacher parameter space (XY plane). LEFT: Without our training paradigm, the teacher is not aware of the student states, the training path and the final state of the teacher can be far away from the student space, i.e. the landscape may be totally different, thus providing limited guidance and lead to the student getting stuck in a local minimum. RIGHT: In our method, the teacher is reset to the student states at the beginning of each round, and does optimization with additional dimensions but within a certain range of the student space, teaching the student with local landscape information and potential direction to a better solution. The number 1∼10 is the step order of these processes, see details in Sec. 4.3.3.

Definition 4.3.1. *Given a model $M^{(A)}_{\theta_A}(I_A)$ (weights θ_A and input I_A), and a model $M^{(B)}_{\theta_B}(I_B)$ (weights θ_B and input I_B), if for any θ_A, there is a θ_B, such that $M^{(A)}_{\theta_A}(I_A) = M^{(B)}_{\theta_B}(I_B)$ for any arbitrary valid input data I_A and its superset I_B. We call model M_B as a* "supermodel" *of M_A.*

The "reset operation" is the process of constructing the weights of supermodel θ_B with the given model weights θ_A, defined as:

Definition 4.3.2. *Given a model $M^{(A)}_{\theta_A}(I_A)$ (weights θ_A and input I_A), and its supermodel $M^{(B)}_{\theta_B}(I_B)$ (weights θ_B and input I_B), we define "reset B with A" to be the process of constructing a new θ_B that meet $M^{(A)}_{\theta_A}(I_A) = M^{(B)}_{\theta_B}(I_B)$ for given θ_A and any valid input I_A and its superset I_B.*

A simple example is, suppose B is a supermodel of A (e.g., $B = A + A'$),

	Accuracy (%) on different angle threshold τ (degree)						
Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**Mean**
Oracle (100% auxiliary modality data)	42.7	68.0	88.0	94.4	96.6	98.6	81.4
one stream (RGB only)	27.3	49.0	77.4	90.2	95.4	98.1	72.9
two streams (shared regressor)	25.9	47.2	77.7	88.4	93.6	97.8	71.8
Modified Xiao et al. [14]	40.8	64.1	84.7	92.7	95.8	98.2	79.4
Modified DMCL [89]	39.1	67.5	88.3	93.9	96.7	98.2	80.6
Ours (AMD-S-Net)	**52.6**	**72.7**	**91.3**	**95.0**	**97.0**	**98.3**	**84.5**

Table 4.1: **Performance comparison for AMD-S-Net under the small amount of auxiliary modality data setting (20%)**.
Our method outperforms other methods by up to **12.7%** mean accuracy improvement.

reset B with A is constructing $\theta_B = [\theta_A, 0]$, where θ_A is the weights of A and 0 is the weights of A'. In Fig. 4.1, the teacher network is a supermodel of the student network, because for any weights of student network, we can construct a teacher network that meet $D(F_{tea}(\{i_n^M, i_n^A\})) = D(F_{stu}(\{i_n^M\}))$ by resetting the F_2 weights with F_4 weights, F_3 weights with F_5 weights, and set F_1 weights to 0. Indeed the reset operation in our method requires that the teacher model is a supermodel of the student model. We also introduce a lemma on the optimal training loss of the supermodel and its base model in Appendix. 4.6.3.

To summarize: (1) The *supermodel* condition ensures the student parameter space is a subspace of the teacher parameter space, thus enable the *reset* operation. (2) The *reset* operation can reset the teacher to be in exactly the same states as the student, which is then utilized by our training paradigm when the teacher gets far from the student, thus allowing the teacher to explore around the student space and teach local landscape information and potential direction of a better solution to the student, achieving superior performance.

4.3.4 Training Paradigm

In this section, we propose a simple yet effective training paradigm based on the "reset operation" (Sec. 4.3.3), which can reset the teacher to exact student states.

As shown in Algorithm. 3, the training paradigm contains t rounds. In each round, we first *reset* the teacher with the student, then train the teacher independently while training the student with both the general label loss and knowledge distillation loss for k epochs. k should not be too large to avoid the teacher being far away from the student. The training process stops when the student converges between different rounds or until finishing t rounds.

Fig. 4.2 shows the training path comparison on loss landscape between general methods and our training paradigm with *reset* operation. Given the teacher network is a *supermodel* of the student network, the student parameter space (along X axis with Y=0) is a subspace of the teacher parameter space (XY plane). Without the *reset* operation, the teacher is not aware of the student states, the training path and the final state of the teacher can be far away from the student space, i.e. the landscape may be totally different, thus providing limited guidance and lead to the student getting stuck in a local minimum (LEFT of Fig. 4.2). In our method, the teacher is reset to the student states at the beginning of each round, and do optimization with additional dimensions but within a certain range of the student space, teaching the student with local landscape information and potential direction to a better solution (right part of Fig. 4.2). Specifically, when the student

is potentially stuck in a local minimum (step 1 in the right part of Fig. 4.2), e.g., already converges with a basic method, we can reset the teacher to the student's states (step 2) and continue to train it (step 3). Then the teacher will be exactly no worse, hopefully better than the student (final position of step 3 is better than the final position of step 1). Then in step 4, which is the distillation training, the student will take both general loss (the force of going downward) and distillation loss (the force of getting closer to the teacher). The distillation loss makes it possible to go upward. After the student pass the loss hill on Y=0, both losses will make it move towards the better solution on Y=0 (final position of step 10).

4.4 Experiments

We first introduce experiment setups in Sec. 4.4.1, then show the results on the real-world dataset in Sec. 4.4.2.

4.4.1 Implementation Details

Setting. All experiments are conducted using one Intel(R) Xeon(TM) W-2123 CPU, two Nvidia GTX 1080 GPUs, and 32G RAM. We use the SGD optimizer with learning rate 0.001 and batch size 128 for training. The number of epochs is 2,000. The loss correlations are $\alpha = 1, \gamma = 1$, while β are set with different values for different knowledge distillation methods following [123], and $\lambda = \beta/10$. We pick epoch number in each round $k = 5$ from ablation study of $k = 1, 2, 5, 20$. We set the round number $n = 400$ for Audi dataset and $n = 40$ for Honda dataset. In the

experiments, each training process is finished within 24 hours.

Evaluation metric. We use the same evaluation metric as a lastest work [152], i.e., the accuracy w.r.t a threshold τ as $acc_\tau = count(|v_{predicted} - v_{actual}| < \tau)/n$, where n denotes the number of test cases; $v_{predicted}$ and v_{actual} indicate the predicted and ground-truth value, respectively. We compute mean accuracy (mAcc) as $\sum_\tau acc_{\tau \in \mathcal{T}}/|\mathcal{T}|$, where $\mathcal{T} = \{1.5, 3.0, 7.5, 15, 30, 75\}$ contains empirically selected thresholds of steering angles.

4.4.2 Results on Real Dataset

We perform main comparisons for our key contributions, i.e., AMD-S-Net, and our training paradigm. We also perform other comparisons on different datasets, modalities, and tasks to show the generalizability of our method, as well as performing comparisons for the robustness of our method. More experiments can be found in the **Appendix PDF** in our project page.

Comparison for AMD-S-Net. Since there's no existing method specifically for the small-shot auxiliary modality distillation, we compare our AMD-S-Net with 2 straightforward frameworks and 2 modified frameworks based on SOTA modality distillation methods. We use Audi dataset [59] and Nvidia PilotNet [58] for this experiment. We use 100% RGB images and 20% segmentation data in this experiment. Specifically, the one stream (RGB only) method uses 100% RGB images only with the student network; two streams (shared regressor) method contains RGB and segmentation pipelines with a feature extractor for each pipeline and a shared

regressor. For modified Xiao et al. [14] and modified DMCL [89], we keep the 20% paired RGB and segmentation to go through the original pipeline, and let the rest 80% RGB data go through a single RGB pipeline. Table. 4.1 shows that our method outperforms other methods by up to **12.7%** mean accuracy improvement.

Combination for our training paradigm. Since our training paradigm can be applied on existing knowledge distillation methods, we conduct experiments by combining ours with kd [94], hint [97], similarity [106], correlation [107], rkd [108], pkt [109], abound [111], factor [112], fsp [113]. One set of experiments use 100% RGB + 100% segmentation, and another set of experiments use 100% RGB + 20% segmentation. From Table. 4.2, our method achieves up to 18.1% improvement in both settings, showing the effectiveness of our training paradigm (containing *reset* operation).

Comparison on different datasets and modalities. We also conduct experiments with different modalities and datasets to show the effectiveness of our method. Specifically, we perform comparison on Audi [59], Honda [60], and SullyChen [24] dataset with RGB image, segmentation, depth map, and edge map modalities. The segmentation is generated by Tao et al. [105], the depth map is generated by [114], and the edge map is generated by DexiNet [115]. Our method outperforms others with up to 11% improvement.

Comparison on different backbones. Except for the Nvidia PilotNet [58], we change the backbone to four other backbones, ResNet [71], ShuffleV2 [153], MobileNetV2 [154], and WRN [155]. Our method outperforms others in all the cases with up to 18.1% improvement.

	Mean Accuracy (mAcc in %)		
Method	$20\%\mathcal{I}^A$	$20\%\mathcal{I}^A$ (ours)	Diff
kd [94]	67.7	73.9	**6.2**
hint [97]	72.7	83.1	**10.4**
similarity [106]	66.4	84.5	**18.1**
correlation [107]	68.5	68.7	**0.2**
rkd [108]	71.2	74.6	**3.4**
pkt [109]	73.4	74.4	**1**
abound [111]	70.6	70.6	0
factor [112]	72.2	84.4	**12.2**
fsp [113]	71.6	71.8	**0.2**
Average	70.5	76.2	**5.7**
Teacher (img+seg)	79.4	-	-
Student (img)	72.9	-	-

Table 4.2: **Performance comparison *with* vs. *without* our training paradigm (containing *reset* operation) under small-shot setting**. By applying our training paradigm on other knowledge distillation methods, we can achieve better performance in most cases (up to **+18.1%**) in fully paired or a small amount of additional modality data.

Comparison on other tasks. Although here we mainly focus on image format auxiliary modalities because it's the most available format, our method can also perform well on other tasks with different data formats, e.g., end-to-end "waypoints prediction task" with point cloud as an auxiliary modality (2.6% improvement), and handwriting classification task with non-image features as auxiliary modalities (2.9% improvement).

Robustness. We also test the robustness of our distilled model following a SOTA work [121] on clean and perturbed Audi dataset (generated with ImageNet-C effects [156]). Our method achieves 4.8% accuracy improvement compared to the RGB only baseline.

4.5 Conclusion

In this chapter, we study the problem of how to introduce a variant amount of auxiliary modality data to increase the performance of single modality learning in an end-to-end steering task. We propose a new framework, **AMD-S-Net**, that can take in the main modality and a variant amount of auxiliary modality data to address this problem. In addition, we propose a novel training paradigm that utilizes *reset* operation to help knowledge transfer. Our AMD-S-Net and training paradigm achieve up to **12.7%** and **18.1%** performance improvement, respectively.

Limitations and Future Work: Our training paradigm assumes that the teacher network is a *supermodel* of the student network. For general knowledge distillation, which usually distills knowledge from a large network to a small network with different architectures, this requirement can possibly limit overall performance gain. However, for modality distillation, when the goal is to reduce the modality instead of reducing the model size, it is common to use a teacher network that has similar architecture as a student network, except for the additional pipeline for auxiliary modalities, as assumed.

Given that it is possible to use a small amount of expert annotation as the auxiliary modality data to improve the performance, what form of expert annotations can be used in the end-to-end steering task or other tasks would be a possible topic for exploration. Also, under the current setting, the auxiliary modality data is *paired with* the main modality data. It is unclear if the same can be applied to unpaired auxiliary modality data to improve the performance, especially without

ground truth.

4.6 Appendix

4.6.1 Supermodel Example

We first introduce the "supermodel" definition:

Definition 4.6.1. *Given a model $M^{(A)}_{\theta_A}(I_A)$ (weights θ_A and input I_A), and a model $M^{(B)}_{\theta_B}(I_B)$ (weights θ_B and input I_B), if for any θ_A, there is a θ_B, such that $M^{(A)}_{\theta_A}(I_A) = M^{(B)}_{\theta_B}(I_B)$ for any arbitrary valid input data I_A and its superset I_B. We call model M_B as a* "supermodel" *of M_A.*

We show a simple example of supermodel in Fig. 4.3. Net_1 contains two blocks f_1 and f_2. Net_2 contains the same block f_1 and f_2, and another block h. If there is a set of specific weights θ_0 for h that can meet $h_{\theta_0}(x) = x$ for any valid x, then Net_2 is a supermodel of Net_1, according to Definition. 4.6.1. In this case, for any specific weights of Net_1, we can always construct a set of weights for Net_2 that has exactly the same performance of Net_1, which means the optimal solution for training Net_2 will be no worse than Net_1. Furthermore, if these two models are training in parallel, the supermodel can be "repositioned" to the same status of the base model at any time by the construction method above. This property can be used in knowledge distillation to let the teacher get back to the student's position and help find a better way at any time the student is stuck.

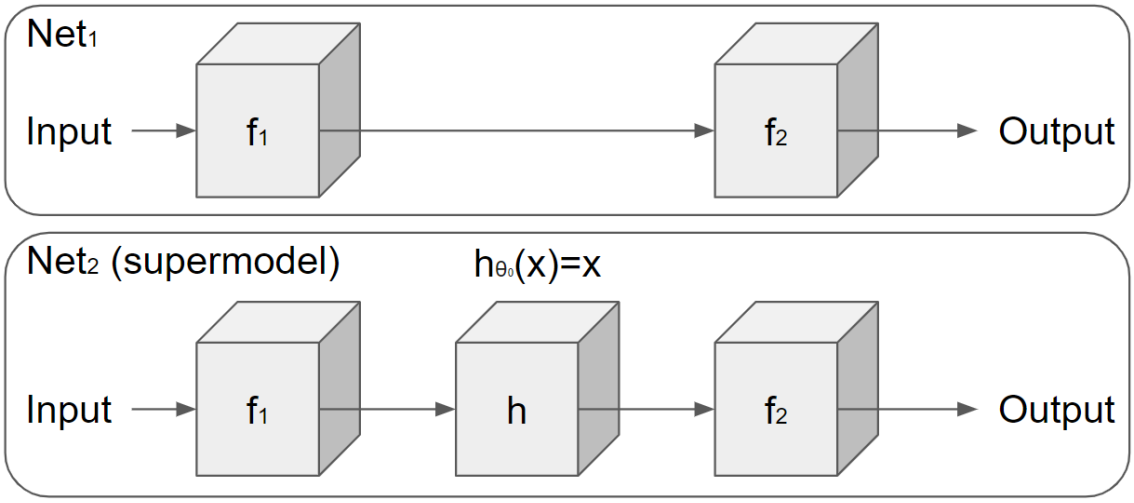


Figure 4.3: A simple example of supermodel. Net_1 contains two blocks f_1 and f_2. Net_2 contains the same block f_1 and f_2, and another block h which is possible to be set as an identical function.

4.6.2 Implementation Details

Setting. All experiments are conducted using one Intel(R) Xeon(TM) W-2123 CPU, two Nvidia GTX 1080 GPUs, and 32G RAM. We use the SGD optimizer with learning rate 0.001 and batch size 128 for training. The number of epochs is 2,000. The loss correlations are $\alpha = 1, \gamma = 1$, while β are set with different values for different knowledge distillation methods following [123] (See details in Appendix 4.6.9), and $\lambda = \beta/10$. We pick epoch number in each round $k = 5$ from ablation study of $k = 1, 2, 5, 20$. We set the round number $n = 400$ for Audi dataset and $n = 40$ for Honda dataset. In the experiments, each training process is finished within 24 hours.

Evaluation metric. We use the same evaluation metric as a lastest work [152], i.e., the accuracy w.r.t a threshold τ as $acc_\tau = count(|v_{predicted} - v_{actual}| < \tau)/n$, where n denotes the number of test cases; $v_{predicted}$ and v_{actual} indicate the pre-

dicted and ground-truth value, respectively. We compute mean accuracy (mAcc) as $\sum_{\tau} acc_{\tau \in \mathcal{T}}/|\mathcal{T}|$, where $\mathcal{T} = \{1.5, 3.0, 7.5, 15, 30, 75\}$ contains empirically selected thresholds of steering angles.

Dataset. For the end-to-end steering task, we do experiments on Audi and Honda datasets. The Audi dataset [59] is the most recent (2020). We use the semantic segmentation subset since it contains both steering angle from bus data and semantic segmentation labels paired with RGB images, which can be used as an additional modality in our method. It contains 41,277 frames in total. The Honda dataset [60] has 100+ long-time driving videos, which is one of the largest autonomous driving datasets. We extract 110k images with 1Hz from the original videos and split them into 100k training images and 10k test images.

Backbone. We choose the Nvidia PilotNet described in [58] as the main backbone. We select this model as it has been used to steer an autonomous vehicle successfully in both the real world [58] and virtual world [48], and also work for the latest autonomous driving datasets [121]. In addition, four other networks are tested to show generalizability.

As shown in Algorithm. 3, the training paradigm contains t rounds. In each round, we first *reset* the teacher with the student, then train the teacher independently while training student with both the general label loss and knowledge distillation loss for k epochs. k should not be too large to avoid the teacher being far away from the student. The training process stops when the student converges between different rounds or until finishing t rounds.

4.6.3 Lemma and Proof

We introduce a lemma on the optimal training loss of the supermodel and its base model.

Lemma 4.6.1. *Given a model M and its supermodel $M^{(s)}$, the optimal training loss of $M^{(s)}$ (which is $\arg\min_{\theta^{(s)}} L(M^{(s)}_{\theta^{(s)}}(I^{(s)}), GT)$) is less than or equal to the optimal training loss of M (which is $\arg\min_{\theta} L(M_\theta(I), GT)$). where L is the loss function and GT is the ground truth.*

Prove: Let $\theta^* = \arg\min_\theta L(M_\theta(I), GT)$ represent the weights that lead to the best training performance for model M, then according to the definition of supermodel, there is a $\theta^{(s)*}$ that meet $M_{\theta^*}(I) = M^{(s)}_{\theta^{(s)*}}(I^{(s)})$, equivalent to $L(M_{\theta^*}(I), GT) = L(M^{(s)}_{\theta^{(s)*}}(I^{(s)}), GT)$. That is, there's at least one solution for training $M^{(s)}$ can get the same performance as training M. Furthermore, if θ^* is the optimal solution that achieves the minimal training loss of $M^{(s)}$, then the equal condition in Lemma. 4.6.1 holds, if not, the less condition holds.

4.6.4 Simple Experiment

We consider a simple task of counting *the number of red circles* in an image of arbitrary shapes of varying colors. In this case, the main modality I^M is the image containing a random layout of arbitrary shapes. We create an auxiliary modality I^A as the images that only contain *red circles*, as shown in Fig. 4.4. For the experiment, we generate 4000 image samples of the main modality with ground-

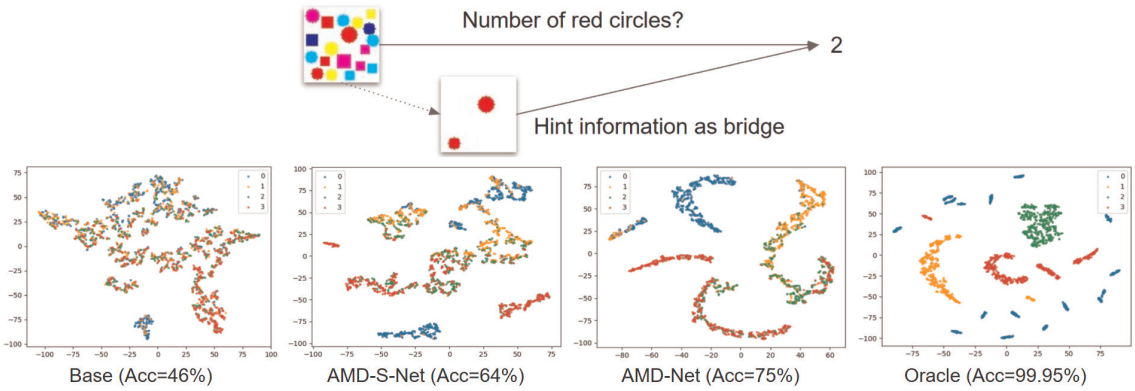


Figure 4.4: Simple experiment explanation. TOP: task definition, counting *the number of red circles* in an image of arbitrary shapes of varying colors, with the images that only contain red circles as the auxiliary modality. BOTTOM: t-SNE visualization for the features generated from networks trained by our methods. The base features are mixed together thus can not be well classified, while our features can be grouped better than the base ones (less mixing points). The oracle features have almost no mixing up in each group because the model has sufficient training data (10x as the base and ours).

truths $\{i_n^M, y_n^M\} \in \mathcal{I}_{train}^M$, and test on another 2000 randomly generated layouts for $\mathcal{I}_{test}^M$. For the auxiliary modality I^A we generate 4000 samples $\{i_n^A\} \in \mathcal{I}^A$. To confirm the hypothesis that the knowledge of I^A can be distilled to improve the task on I^M, we design an ablation study comparing the following baselines:

- Oracle $\mathcal{I}^M$: $\mathcal{I}_{train}^M$ is sufficient for θ^{M*};
- Underfitted $\mathcal{I}^M$: $\mathcal{I}_{train}^M$ is not sufficient for θ^{M*};
- Underfitted $\mathcal{I}^M$ plus auxiliary $\mathcal{I}^A$: Add auxiliary samples from $\mathcal{I}^A$ to insufficient $\mathcal{I}^M$.
- Underfitted $\mathcal{I}^M$ plus insufficient auxiliary $\mathcal{I}^A$: Add auxiliary but a small number of samples from $\mathcal{I}^A$ to insufficient $\mathcal{I}^M$.

For the *Oracle* experiment, all 4,000 training samples are used, the model is nearly

perfect at inference, achieving 99.95% accuracy on $\mathcal{I}^M_{test}$. In an *Underfitted* situation, we randomly sample 10% from $\mathcal{I}^M_{train}$ and have created an insufficiently trained classifier with 46% accuracy. After that, in addition to the *Underfitted* model we add 400 auxiliary samples from $\mathcal{I}^A$, the accuracy improves to 75% (29 percent improvement). Even without sufficient auxiliary samples, when we select merely 80 samples from $\mathcal{I}^A$ and use AMD-S-Net, we still witness a plausible improvement to 64% (18 percent improvement). We generate the t-SNE visualizations for the representations from each baseline in Fig. 4.4, and observe clearly enhanced clusters with the distilled knowledge from auxiliary modality achieved by our methods.

4.6.5 Modifications on SOTA Frameworks

We compare our framework with 2 straightforward frameworks and 2 modified frameworks based on SOTA modality distillation methods. We use 100% RGB images and 20% segmentation data in this experiment. Specifically, the one stream (RGB only) method uses 100% RGB images only with the student network. Two streams (shared regressor) method contains RGB and segmentation pipelines with a feature extractor for each pipeline and a shared regressor. The total loss is the sum of RGB loss and segmentation loss. During test, only the RGB pipeline is used. For the modified [14] and modified DMCL [89], we keep the 20% paired RGB and segmentation to go through the original pipeline with backpropagation, and let the rest 80% RGB data go through a single RGB pipeline with backpropagation. During test, only the RGB pipeline is used.

4.6.6 Comparison with SOTA Knowledge Distillation Methods

We show the detailed comparison data our AMD-S-Net in Table 4.10. With our method, the performance improvement can get up to 18.1%.

4.6.7 Multi-Modal End-to-End Waypoint Prediction

To show the generalizability of our method, we do experiments on another end-to-end autonomous driving task, way points prediction task. Following the setting of [117], we consider the task of navigation along a set of predefined routes in different areas, such as motorways, urban regions, and residential districts. A sequence of sparse goal locations in GPS coordinates provided by a global planner and the related discrete navigational commands, such as "follow lane", "turn left/right", and "change lane", constitute the routes. Only the sparse GPS locations are used in our method. Each route is constituted of several scenarios that are initialized at predefined locations and test the agent's ability to handle various adversarial situations, such as obstacle avoidance, unprotected turns at intersections, vehicles running red lights, and pedestrians emerging from occluded areas crossing the road at random locations. The agent needs to complete the route within a certain amount of time, while following traffic restrictions and dealing with large numbers of dynamic agents. For dataset, we use the CARLA [125] simulator for training and testing, specifically CARLA 0.9.10 which consists of 8 publicly available towns. We use 7 towns for training and hold out Town05 for evaluation as in [117]. See Table. 4.3.

Model	DS↑	RC↑	IP↓	CP↓	CV↓	CL↓	RLI↓	SSI↓
RGB	21.0	60.5	0.49	**0.01**	0.15	0.08	**0.14**	0.04
RGB+PC	11.2	52.9	**0.37**	0.02	0.22	0.01	0.38	**0.02**
Ours	**22.0**	**63.1**	0.45	0.02	**0.05**	**0.00**	0.20	0.03

Table 4.3: Performance comparison on long routes way points prediction between base (100% RGB), multi-modality (28% RGB + 28% point cloud), and our method (100% RGB + 28% point cloud). DS: Avg. driving score, RC: Avg. route completion, IP: Avg. infraction penalty, CP: Collisions with pedestrians, CV: Collisions with vehicles, CL: Collisions with layout, RLI: Red lights infractions, SSI: Stop sign infractions.

4.6.8 Handwriting Classification

We also perform comparison on multi-feature handwritten classification task [116] in Table. 4.4. The dataset [124, 157, 158] consists of six features of handwritten numerals ('0'–'9') with 2000 samples in total. We regard the six feature sets as six modalities, and treat each of them as target modality in each experiment. Teacher network is able to get all 6 modalities (but only 20% amount of data). During test, only one target modality is available. Our method outperforms others with 2.9% in average.

	Accuracy (%) on different modalities (ID:1∼6)						
Method	1	2	3	4	5	6	mean
Other KD	84.92	**62.98**	68.75	61.10	70.35	43.17	65.2
Ours	**87.42**	62.29	**70.86**	**66.34**	**71.97**	**49.49**	**68.1**

Table 4.4: **Performance comparison on handwritten classification task.** Our method outperforms other KD methods with 2.9% on average.

				Accuracy (%) on different angle threshold τ (degree)						
Dataset	Train Mod	Test Mod	Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**mAcc**
Audi	RGB+seg	RGB+seg	Teacher	42.7	68.0	88.0	94.4	96.6	98.6	81.4
Audi	RGB+seg	RGB	best others	30.3	51.0	78.2	88.4	94.4	98.2	73.4
	RGB+seg	RGB	ours	**52.6**	**72.7**	**91.3**	**95.0**	**97.0**	**98.3**	**84.5**
Audi	RSDE	RSDE	Teacher	49.9	72.1	89.5	94.9	97.1	98.6	83.7
Audi	RSDE	RGB	best others	27.7	47.8	77.4	90.8	95.6	98.3	72.9
	RSDE	RGB	ours	**30.2**	**50.3**	**79.7**	**91.0**	**96.2**	**98.6**	**74.3**
SullyChen	RDE	RDE	Teacher	41.1	63.7	88.6	95.9	97.9	99.1	81.0
SullyChen	RDE	RGB	best others	59.5	82.1	93.9	98.2	99.5	100.0	88.9
	RDE	RGB	ours	**63.4**	**83.0**	**94.3**	98.2	99.5	100.0	**89.7**
Honda	RSDE	RSDE	Teacher	41.3	61.1	83.9	94.0	98.3	99.9	79.8
Honda	RSDE	RGB	best others	**38.9**	**57.7**	79.7	91.7	97.5	99.3	77.4
	RSDE	RGB	ours	37.9	**57.7**	**81.7**	**93.5**	**98.2**	**99.6**	**78.1**

Table 4.5: Comparison on different datasets and different modalities. RSDE is short for RGB + segmentation + depth map + edge map, and RDE is short for RGB + depth map + edge map. Our method outperforms others on different datasets and different additional modalities with up to 11% accuracy improvement.

4.6.9 Knowledge Distillation Methods Settings

For different knowledge distillation methods, different values of β (weight of the consistency loss) is used. We use the same setting as [123]. Specifically:

- kd [94]: $\beta = 0$
- hint [97]: $\beta = 100$
- similarity [106]: $\beta = 3000$
- correlation [107]: $\beta = 0.02$
- rkd [108]: $\beta = 1$
- pkt [109]: $\beta = 30000$
- abound [111]: $\beta = 1$

		Accuracy (%) on various angle threshold τ (degree)				
Backbone	Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	**mAcc**
PilotNet	SIM	20.6	38.9	66.7	81.5	66.4
PilotNet	SIM+ours	**52.6**	**72.7**	**91.3**	**95.0**	**84.5**
ResNet34	SIM	30.1	54.4	85.5	**94.1**	76.6
ResNet34	SIM+ours	**37.2**	**60.2**	**85.7**	93.3	**78.6**
ShuffleV2	SIM	39.9	61.3	81.4	89.8	77.7
ShuffleV2	SIM+ours	**47.0**	**71.2**	**90.1**	**94.9**	**83.0**
MobileNetV2	SIM	31.1	51.4	78.2	89.4	73.9
MobileNetV2	SIM+ours	**52.9**	**71.8**	**89.7**	**94.6**	**84.0**
WRN	SIM	22.8	42.9	76.9	92.2	71.7
WRN	SIM+ours	**37.7**	**64.7**	**89.8**	**94.6**	**80.3**

Table 4.6: Performance comparison on different backbones. Our method outperforms SIM [106] on PilotNet [58], ResNet34 [71], ShuffleV2 [153], MobileNetV2 [154], and WRN [155] with up to 18.1% accuracy improvement.

- factor [112]: $\beta = 200$
- fsp [113]: $\beta = 50$
- attention [142]: $\beta = 1000$

4.6.10 Comparison on Different Datasets and Modalities

We also do comparison with other knowledge distillation methods on different datasets (Audi [59], Honda [60], and SullyChen [24]) and different modalities (RGB, segmentation, depth map, and edge map). Specifically, Audi dataset contains ground truth segmentation, and other segmentation is generated by Tao et al. [105], while the depth map is generated by [114] and the edge map is generated by DexiNet [115]. In Table. 4.5, our method outperform others in all cases with up to 11% accuracy improvement.

	Accuracy (%) on various angle threshold τ (degree)				
Type of I^A	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	**mAcc**
Base (No I^A)	28.3	49.5	79.7	89.1	73.1
Depth map [114]	33.3	57.6	81.5	90.4	75.9
Edge map [115]	34.9	56.2	79.6	90.9	76.0
Segmentation [105]	**35.2**	**58.3**	**83.3**	**91.6**	**77.1**

Table 4.7: Comparison on different types of auxiliary modalities on Audi dataset, with a basic L2-norm feature loss for the knowledge distillation process. We show that all the auxiliary modalities can perform better than the base model by at least 2.8%. This shows our algorithm can utilize different types of auxiliary modalities well, even with a basic knowledge distillation loss.

4.6.11 Comparison on different backbones.

Except for the Nvidia PilotNet [58], we change the backbone to four other backbones, ResNet [71], ShuffleV2 [153], MobileNetV2 [154], and WRN [155], and do comparison in Table. 4.6. Our method outperforms other methods in all the cases with up to 18.1% accuracy improvement.

4.6.12 Comparison on different tasks.

4.6.13 Effectiveness on Different Modalities

We do comparison on different types of auxiliary modalities on Audi dataset in Table. 4.7, with a basic L2-norm feature loss for the knowledge distillation process. We show that all the auxiliary modalities can perform better than the base model by at least 2.8%. This shows our algorithm can utilize different types of auxiliary modalities well, even with a basic knowledge distillation loss.

	Clean	Blur				Noise			
	Clean	Defocus	Glass	Motion	Zoom	Gauss	Shot	Impulse	
RGB only	73.1	72.7	71.8	69.8	72.3	67.9	66.9	67.0	
$20\%\mathcal{I}^A$	74.8	74.3	73.1	73.2	74.2	69.2	68.3	68.6	
$100\%\mathcal{I}^A$	77.1	75.5	75.2	73.1	76.3	71.4	70.1	70.3	
	Clean	Weather				Digital			**mAcc**
	Clean	Snow	Frost	Fog	Bright	Contrast	Pixel	JPEG	**mAcc**
RGB only	73.1	62.8	56.5	54.2	64.2	39.9	73.3	70.7	**65**
$20\%\mathcal{I}^A$	74.8	68.1	65.4	63.8	67.6	65.4	74.8	71.8	**69.8**
$100\%\mathcal{I}^A$	77.1	63.8	58.7	56.4	65.8	62.0	77.2	75.3	**69.4**

Table 4.8: Average accuracy(%) of our method on clean and perturbed data (generated with ImageNet-C effects [156]). The last column "Mean" is the mean accuracy on all perturbed data (blur, noise, weather and digital). We show that both basic and small-shot auxiliary modality learning can get higher accuracy than the base method (about 4.7% in average), i.e., higher robustness.

	Accuracy (%) on different threshold τ (degree)						
Input	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**Mean**
RGB	32.4	53.2	78.7	87.7	94.1	97.8	74.0
RGB + 3 Rand	30.3	51.7	79.8	88.4	94.4	97.5	73.7

Table 4.9: RGB image plus 3 random channels as input can perform nearly as well as only RGB image as input, showing adding useless channels will not influence the performance too much.

4.6.14 Robustness

We test the robustness of our distilled model following a SOTA work [121] on clean and perturbed Audi dataset (generated with ImageNet-C effects [156]). Table. 4.8 shows our method can also improve the robustness while not seeing any of the perturbed images.

4.6.15 Random auxiliary data

When use random noisy as auxiliary modality, our method will not be affected, see Table. 4.9.

	Accuracy on different threshold τ (%)							
Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**Mean**	**Improvement**
train vanilla								
Teacher (img+seg)	40.8	64.1	84.7	92.7	95.8	98.2	79.4	
Student (img)	27.3	49.0	77.4	90.2	95.4	98.1	72.9	
existing distillation methods								
kd [94]	23.4	41.2	68.9	83.7	92.1	97.2	67.7	
hint [97]	28.3	47.6	77.8	89.2	95.0	98.4	72.7	
similarity [106]	20.6	38.9	66.7	81.5	92.6	98.0	66.4	
correlation [107]	21.7	39.5	70.0	86.8	94.6	98.2	68.5	
rkd [108]	26.2	46.5	74.8	87.9	94.1	97.8	71.2	
pkt [109]	30.3	51.0	78.2	88.4	94.4	98.2	73.4	
abound [111]	24.8	45.2	74.9	87.3	93.7	97.7	70.6	
factor [112]	26.8	47.8	76.9	88.8	94.7	98.0	72.2	
fsp [113]	27.1	47.7	74.4	87.9	94.4	97.8	71.6	
attention [142]	27.1	47.0	73.1	84.9	92.8	98.3	70.5	
existing distillation methods with our training paradigm								
kd [94]	30.4	53.7	78.5	88.3	94.8	97.8	73.9	**6.2**
hint [97]	52.7	71.2	88.8	93.6	95.5	97.1	83.1	**10.4**
similarity [106]	52.6	72.7	91.3	95.0	97.0	98.3	84.5	**18.1**
correlation [107]	21.7	39.7	71.2	87.0	94.4	98.2	68.7	**0.2**
rkd [108]	32.4	53.8	79.5	89.3	94.7	97.9	74.6	**3.4**
pkt [109]	54.2	72.5	90.0	94.8	96.7	98.3	84.4	**11**
abound [111]	24.9	45.3	75.1	87.1	93.5	97.7	70.6	0
factor [112]	54.3	72.3	90.1	94.8	96.7	98.3	84.4	**12.2**
fsp [113]	27.5	48.4	75.0	87.5	94.3	97.8	71.8	**0.2**
attention [142]	46.2	68.1	86.8	93.4	96.6	98.2	81.5	**11**

Table 4.10: Comparison with knowledge distillation methods on Audi dataset (100% RGB image + 20% segmentation) with Nvidia PilotNet [58]. First section in the table shows the performance of teacher and student network trained directly. Second section shows the performance of student with different knowledge distillation methods (train student from start, using the pretrained teacher model in the previous section). Third section shows the performance of student after using our technique based on other methods (take the teacher and student network in the second section of this table as init model, and retrain the model with our method). By comparing between the second and third section, we can see our method increase the performance of most existing methods with up to 18.1%.

Chapter 5: Task-Driven Domain-Agnostic Learning for Autonomous Steering

5.1 Introduction

Autonomous driving (AD) has the potential to create safer and more efficient transportation systems by reducing congestion and accidents due to human errors. Central to AD, autonomous steering is a complex task and requires the choreography of many components to operate. One essential component is the perception-control module that maps sensor data to control commands (e.g., steering angles). With recent advances in machine learning, especially deep learning [22], the perception-control module is increasingly enabled by learning-based algorithms, which leverage multimodal input from sensors including cameras, Lidar, and radar to navigate autonomous vehicles (AVs). While each type of sensor offers its unique strength in detecting the environment, the camera is one of the most universal and accessible sensors due to its rich visual information and affordable cost.

As a result, many real-world images are collected for training AVs. Example datasets include KITTI [23], NVIDIA [24], Waymo Open Dataset [25], CityScapes [26], and BDD100K [27]. In addition to real-world images, simulators and virtual images

are also heavily used in training AVs [28]. Example simulation platforms include CARLA [29], the Udacity Self-Driving Car Simulator [30], and NVIDIA Drive Constellation [31]. Many scenarios that are crucial for testing autonomous driving but difficult to capture in the real world can be modeled in the virtual world at ease, e.g., accidents. While it is believed that virtual images can supplement real images, the domain gap between the two can obstruct the conjecture. Furthermore, the recent advancement of image style transfer techniques [38] such as CycleGAN [39] and MUNIT [40] challenges the domain gap and has raised new conjectures on whether we can use the realistic-looking images converted from virtual images for learning [41]. In this work, we explore not only the domain gap between virtual and real images but also style-transferred images, in order to understand how the domain gap and style-transfer techniques influence the performance of "learning to steer". We also analyze how different training paradigms can reduce the domain gap, e.g., finetuning, partially finetuning, and finetuning with reinitialization. In addition, another common way to reduce the domain gap is modifying the network architecture, especially with certain additive network components for easy modification, e.g., Batch Norm layers or Adapters. We investigate normal BN layers, AdvProp BN, and LoRA Adapter. Finally, we show the transferability will vary under different amounts of target data. See analysis details in Sec. 5.4.

Based on the analysis, we propose a novel framework for domain-agnostic learning in the steering task, i.e., improve the target domain performance with additional source domain data. We analyze the impact of three key components: network architecture, training data, and training paradigm in autonomous steering.

Specifically, we use (1) domain-specific adapters and shared modules to disentangle *domain-specific* information and *task-specific* information; (2) style-transferred branch to help extract domain-specific information; (3) gradually increased ratio of target domain data in each epoch for better knowledge transfer from source to target domain. See detail of our framework in Sec. 5.5.

Overall, the main contributions of this work include:

- Analyze how different factors influence the end-to-end steering task, including training data (image style, data amount from source and target domain), network architecture (Batch Norm layers, Adapters), and training paradigm (finetuning, partially finetuning, reinitialization).

- Propose a novel framework to solve domain-agnostic learning in the end-to-end steering task, with specific design from training data, network architecture, and training paradigm perspectives.

5.2 Related Work

5.2.1 Transfer Learning

Transfer learning is a problem that has been studied for years. There are different types of transfer learning according to different settings. Task adaptation is one of them when target domain data labels are available. LWF [32] is able to learn in the target domain while keeping the memory of the source domain without source domain data. DELTA [33] proposes a novel regularized transfer learning

framework, preserving the outer layer outputs of the target network. BSS [34] presents a novel regularization approach to penalizing smaller singular values so that untransferable spectral components are suppressed. StochNorm [35] proposes a two-branch design with one branch normalized by mini-batch statistics and the other branch normalized by moving statistics. Co-Tuning [36] is a two-step framework that can learn the relationship between source categories and target categories, and use source and target labels to collaboratively supervise the fine-tuning process. Bi-Tuning [37] presents a general learning framework to fine-tune both supervised and unsupervised pre-trained representations to downstream tasks.

Compare to them, our method is among the first that considers three perspectives, i.e., training data, training paradigm, and network architecture, while most previous work only considers one or two perspectives.

5.2.2 Virtual and Real Data

While collecting real-world data can be expensive and challenging, the virtual world enables the economical production of a large amount of data. In order to study how virtual images influence learning-based tasks, researchers have adopted various style-transfer techniques. For example, Movshovitz-Attias et al. [159] explore the effect of state-of-the-art rendering techniques on the viewpoint estimation task of objects. Another style-transfer technique, Generative Adversarial Networks (GANs), has been used for domain transfer between different types of images [160]. Among many variants of GAN, CycleGAN [39] has been successfully applied to

style-transfer unpaired images in training data. CyCADA [161], a follow-up work of CycleGAN, improves the performance of adversarial adaptation models by preserving local structural information as well as semantic consistency. However, CyCADA does not improve the visual realism of converted images. Supplementing virtual images with unlabeled real images has been shown to improve the quality of GANs' output. As an example, Shrivastava et al. [162] propose "simulated + unsupervised" learning, which aims to improve learning performance on large datasets without extra data collection or annotation efforts. They train a GAN-similar refiner network, called SimGAN, to create realistic photos without annotated real photos. Finally, the blending of virtual world and real world has shown potential for learning-based driving tasks. Li et al. [163] proposed an augmented autonomous driving simulation (AADS), which introduces simulated traffic flows into real-world environments. The training environment is obtained by scanning the real world with lidar and cameras, while simulated traffic flows, including vehicles and pedestrians, are mapped onto the scanned environment. This method captures the benefits of a fully-controlled virtual environment, while retaining realism.

In contrast to the above-mentioned studies, we explore combining virtual, style-transferred, and real images in various proportions and for different training strategies. We then base our experiments in studying the influence of these settings on the performance of the task "learning to steer" an autonomous vehicle.

5.3 Problem Setting

Problem Description. A major challenge for autonomous driving is the variety of driving scenarios, because it is impossible to train using data from all possible scenarios. When we encounter a new scenario, we should train a good model with both existing data in known scenarios (source domain) and new data (target domain), which can perform better than the model trained by only the new data.

Base Datasets. We use the Nvidia dataset [24] as our real dataset, which contains approximately 63,000 images at the resolution 455 $\times$ 256. The data is recorded on urban/suburban roads in California. We use the data from the Udacity Self-Driving Car Simulation [30] as our virtual dataset, which includes about 10,615 images at the resolution 320 $\times$ 160 and is collected on a simulated suburban driving track. In order to make the size of the two datasets equal, we randomly select 10,615 images from the Nvidia dataset as the final real dataset.

While both datasets are similar in visual contents, the definition of the labels differ. In the Nvidia dataset, the label is the steering angle, while in the Udacity dataset the label is the turning angle of the front wheels. We convert the labels in the Udacity dataset into steering angles by scaling them up by 15.06, which is the steering-to-turning ratio of the 2014 Honda Civic [164], the vehicle used to collect the Nvidia dataset. The maximum steering angle is 338 degrees. Sample images of the two datasets can be found in Fig. 5.2(a) and Fig. 5.2(b), respectively.

Evaluation Metric. We use mean accuracy (MA) to evaluate our regres-

sion task, since it can represent the overall performance under different thresholds. We first define the accuracy with respect to a particular threshold τ as $acc_\tau = count(|v_{predicted} - v_{actual}| < \tau)/n$, where n denotes the number of test cases; $v_{predicted}$ and v_{actual} indicate the predicted and ground-truth value, respectively. Then, MA is computed as $\sum_\tau acc_{\tau \in \mathcal{T}}/|\mathcal{T}|$, where $\mathcal{T} = \{1.5, 3.0, 7.5, 15, 30, 75\}$ contains empirically selected thresholds of steering angles.

Backbone. We choose the model by Bojarski et al. [58] as the default backbone. The model contains five convolutional layers followed by three dense layers. We select this model because it has been used to steer an AV successfully in both real world [58] and virtual world [48].

Notation and Training Strategies. In this work, we explore different ways to combine datasets for learning, in order to determine their potential to improve learning performance. We define the following notation for any two datasets A and B.

- train($A+B$): simply combine datasets A and B and use the combined dataset for training;
- train(A) $\rightarrow$ train(B): use A to pretrain a model, then use B to retrain the model; and
- train(A) $\rightarrow$ ptrain(B): use A to pretrain a model, then use B to retrain the model by only updating partial weights of the model. This training strategy is inspired by transfer learning [165]. Since we are using the model by Bojarski et al. [58], we only update the weights in the fully connected layers during

the retraining of the model using dataset B, while keeping the weights in the convolutional layers learned using dataset A. This operation is based on the assumption that the convolutional layers can extract domain-invariant, low-level features across different categories of images.

The output of the above-mentioned three training methods is a learned model. We use $\mathrm{MA}_A(M)$ to denote the MA score of testing a learned model M using the test set extracted from dataset A. For all datasets, we split them into the training set and test set using the ratio 10:1.

5.4 Analysis

In this section, we analyze how training data (image style, data amount), network architecture (Batch Norm layers, Adapters), and training paradigm (fine-tuning, partially finetuning, reinitialization) influence the transfer learning.

5.4.1 Does Image Style Transfer Reduce Domain Gap?

To reduce the gap between two domains, the first intuition is to improve the visual similarity of the training data. In this regard, we study how image style can influence the end-to-end steering task.

Some existing works can change the style of an image set to the style of another image set. We use two style-transfer algorithms in this work, CycleGAN [39] and MUNIT [40]. We show sample images of the two types of style-transferred images in Fig. 5.2(c) and (d), respectively.

Training Dataset	R	V	T_C	T_M
$\text{MA}_R(M)$	88.36%	31.16%	26.87%	25.56%

Table 5.1: Test Mean Accuracy of models trained on real (R), virtual (V), virtual to real with CycleGAN (T_C), virtual to real with MUNIT (T_M). **Transferring the image style** with the two learning-based methods could **not reduce the gap** between virtual and real domain in the steering task.

- R: real dataset (the Nvidia dataset [24], target domain);
- V: virtual dataset (the Udacity dataset [30]);
- T_C, T_M: style-transferred datasets from virtual to real using CycleGAN [39] and MUNIT [40], respectively.

All four datasets R, V, T_C, T_M contain the same number of (i.e. 10K) images. We start by exploring the domain gap between the three types of datasets. The results are shown in Table 5.1. Our first finding is that domain gaps exist between the virtual and style-transferred images, as well as style-transferred and real images. Using the pair R and T_C as an example, the difference of the corresponding MA values, $\text{MA}_R(\text{train}(R))$ - $\text{MA}_R(\text{train}(T_C))$ = 88.36% - 26.87% = 61.49%, indicating the existence of a domain gap between style-transferred and real images.

Our second finding is that although style-transferred images can sometimes look more "real" than virtual images (in Fig. 5.2 (c) and (d)), they are not necessarily "closer" to real images than virtual images in a learning task. This is reflected by the result shown in Table 5.1(a): $\text{MA}_R(\text{train}(T_C))$ ¡ $\text{MA}_R(\text{train}(V))$ ¡ $\text{MA}_R(\text{train}(R))$, and $\text{MA}_R(\text{train}(T_M))$ ¡ $\text{MA}_R(\text{train}(V))$ ¡ $\text{MA}_R(\text{train}(R))$. This indicates the learning-based style transfer method may include additional domain gap

factors for the steering task when it tries to improve visual similarity.

We then replace the MUNIT method with a traditional *color remapping* method, i.e., map the distribution of RGB values in the virtual domain to the real domain. Although the images generated by color remapping method is not as real as the learning-based methods (See Appendix. 5.7.1), the *test accuracy is better*, i.e., 30.08%.

We then do cross comparison on six domains, R, V, RV_{CGAN}, VR_{CGAN}, RV_{CR}, and VR_{CR}, which is a combination of real/virtual content + real/virtual style (+ learning/non-learning-based method). We train models on each of them separately, then test on them separately. Table 5.2 shows the cross comparison results. We found:

- (a) **Image content is more important than image style**. When testing on real-content datasets (R, RV_{CGAN}, RV_{CR}), the models trained on real-content datasets perform better than the models trained on virtual-content datasets, no matter they are real or virtual style (the bolden numbers are greater than the unbolden numbers).

- (b) *With the same content during training and test (the bolden numbers), using same style is better than using different styles* (the diagonal of the bolden numbers are greater than other numbers).

- (c) With different content during training and test (the unbolden numbers), same style is not necessary to perform better (when testing on R, the model trained on V performs best but they are not in the same style, similar for

Test	Train					
	R	V	RV_{CGAN}	VR_{CGAN}	RV_{CR}	VR_{CR}
R	**88.36%**	*31.16%*	**48.83%**	26.87%	**70.17%**	30.08%
RV_{CGAN}	**51.42%**	34.22%	**80.08%**	29.34%	**53.18%**	*38.86%*
RV_{CR}	**60.89%**	35.86%	**48.18%**	27.79%	**85.50%**	*37.41%*

Table 5.2: Mean Accuracy cross comparison. RV stands for transferring real dataset to virtual style, VR stands for transferring virtual dataset to real style. $CGAN$ stands for the Cycle-GAN method, and CR stands for the color remapping method.

testing on RV_{CGAN} and RV_{CR}).

In addition, we try to evaluate the domain gap with Fréchet Inception Distance (FID) [64], so that we don't need to train models when we met new domains. However, we found that **FID is not necessary an effective metric for evaluating the domain gap in steering task**. As shown in Table 5.8 (in Appendix 5.7.2), the relative order is different from the actual test results in Table 5.2.

5.4.2 Training Paradigm

The most common technique in transfer learning to cross the domain gap is modifying the training paradigm, i.e., changing the way of training without modifying the network architecture. Popular methods [166] include,

- Finetuning. Retrain a model on the target domain which is pretrained on the source domain.
- Partially finetuning. Finetuning a model with fixed weights of specific layers, e.g., CNN layers.

- Finetuning with reinitialization. Reinitialize specific layers before retraining. In our experiments, we use header reinitialization.

Table 5.3 shows the Mean Accuracy comparison with different training paradigms and source domains. From (a) we can verify the existence of a domain gap between the virtual, style-transferred and real datasets. (b) is a basic paradigm that trains a model with source and target domain data simply combined. From (b,c,d,e,f), we find that **(e)"finetuning with reinitialization" outperforms other training paradigms in the list**, no matter using real ($R1$, Audi dataset [59]), virtual (V), or style transferred (T_C, T_M) datasets as source domain.

5.4.3 Network Architecture

Except for the training paradigm related methods, there are methods that achieve transfer learning by modifying the network architecture, e.g., batch norm layers, adapters, etc. Usually they also need to have specific training paradigms, e.g., adding batch norm layers, retrain the model with fixed CNN layers but trainable batch norm weights. Here we mainly investigate two additive network components,

- **Batch Norm (BN)** layer [167]. Batch normalization is a method used to make training of artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling. Different domains have different feature distributions, which can be aligned by adding batch norm layers in the network.

- **Adapter** [168]. Adapters are new modules added between layers of a pre-trained network. they add only a few trainable parameters per task, and new tasks can be added without revisiting previous ones. The parameters of the original network remain fixed, yielding a high degree of parameter sharing.

In Table 5.4, we compare normal BN [167], AdvProp BN [169], and LoRA [170] under the best training paradigms in previous experiments. **LoRA achieves the best performance in the list.**

5.4.4 Other Factors

In addition to the factors above, the transferability will also be influenced by other factors, e.g., data amount, loss function, etc. We also do explorations of the data amount and show our findings here. For data amount, *a hypothesis is that the transferability will vary under different amounts of target data during training*. When the target domain data is adequate, it's difficult to further improve the performance with additional source domain data. However, *when the target domain data is insufficient, then adding source domain data may help the model learn better*. Table 5.5 (e) (f) verifies this hypothesis.

5.5 Our Method

Inspired by the analysis in Sec. 5.4, we design our framework from three perspectives, i.e., training data, network architecture, and training paradigm.

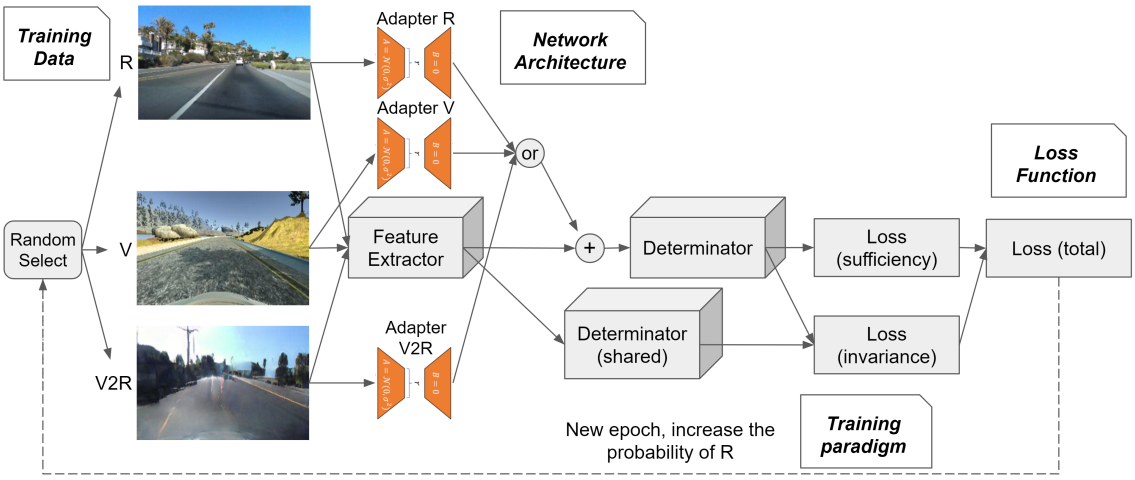


Figure 5.1: **Our framework**: In each epoch, the input data is randomly selected from multiple domain data, which will be fed into a shared feature extractor and a domain-specific adapter for each domain. The combined output feature will then be used to determine the final steering angle (sufficiency loss). An additional invariance loss is used to force the feature extractor to extract as much domain-invariant information as possible. The final loss contains both sufficiency loss and invariance loss. The initial probability of the target domain is small, but will gradually increase after each epoch, to better transfer knowledge from source to target domain.

5.5.1 Framework

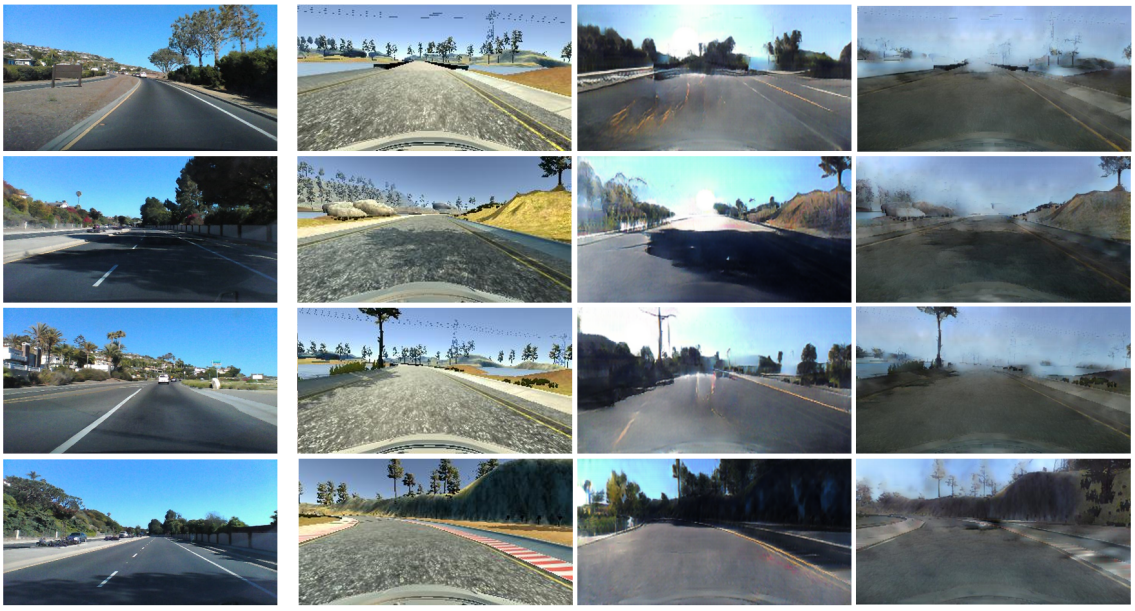
Overview. Our framework overview is shown in Fig. 5.1. In each epoch, the input data is randomly selected from real, virtual, and style-transferred datasets (according to a probability variable for each of them), which will be fed into a shared feature extractor and a domain-dependent adapter. The combined output feature will then be used to determine the final steering angle. The initial probability of the target domain R is small, but will gradually increase after each epoch, to better transfer knowledge from source (V) to target (R) domain.

Design in network architecture. As shown in Sec. 5.4.1, different image styles between the source and target domain hurt the performance. Then we consider to align, or remove the styles. Out of expectation, experiments show that

transferring the image style from source to target domain with existing style transfer methods fails to bring benefits. Thus we consider the removal of the styles. We use a shared feature extractor to deal with image content, and domain-dependent adapters to deal with the image styles. The intuition is, both real and virtual domain share common information in this steering task, e.g., front-end perception, or back-end steering control, and the common part is supposed to be dealed with the shared modules like feature extractor and determinator. Other than the shared part, the adapter is supposed to extract the domain-specific information, e.g., image style. A previous work AdvProp BN [169] uses independent Batch Norm layers for different domains, but the domain-specific information is not necessary to be held uniformly in one branch with just different BN parameters, similar problem for StochNorm [35].

Design in training data. The style transferred branch is added to the framework to better separate the image style and image content. Inspired by recent works [171, 172], we can add "hint" images in the input as prior information to help the model learn better. Originally we only have real content + real style, and virtual content + virtual style. With style-transferred images, i.e., real content + virtual style, or virtual content + real style, the model is able to get more hints about the borderline of content and style information. The style transferred data can be generated by CycleGAN [39], a generative model which can exchange the image style of two sets of unpaired images by using a forward and backward supervision. See examples in Fig. 5.2.

Design in training paradigm. In Sec. 5.4.2, we show that **finetuning is**



(a) Real (R) (b) Virtual (V) (c) CycleGAN (T_C) (d) MUNIT (T_M)

Figure 5.2: Sample images of various datasets. (a) the Nvidia dataset [24] (real dataset, denoted by R). (b) the Udacity dataset [30] (virtual dataset, denoted by V). (c) style-transferred images from virtual to real using CycleGAN [39] (denoted by T_C). (d) style-transferred images from virtual to real using MUNIT [40] (denoted by T_M). We plan to release all datasets for comparative experiments.

better than simply combining two datasets. An explanation for this phenomenon is, *learning two skills together is harder than learning one skill first and then another*. In our architecture, since we want to split the image content and style by feeding the two domain data together, there is no training and retraining step. Thus we use a probability variant for each domain to control the learning process, e.g., learn the source domain first, then gradually increase the target domain data. Experiments show this strategy is better than simply using a same number of data from each domain in one epoch.

Loss Function. The loss function contains two parts, sufficiency loss and invariance loss. For sufficiency loss, it's a straightforward L2 loss between the output

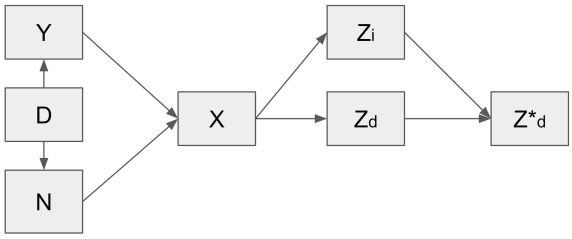


Figure 5.3: A Structural Causal Graph that shows causal relationships between variables. We decouple the randomness in the input variable X into the task label Y and random variable nuisance N, both are dependent on the domain D. We extract domain-invariant latent variable Z_i and domain-dependent latent variable Z_d from input X, then combine Z_i and Z_d to form a compressed latent variable Z_d^*.

of the network and the ground-truth steering angle. The invariance loss is used to force the feature extractor to extract as much domain-invariant information as possible. See more details in Sec. 5.5.2.

5.5.2 Information Theoretic View of Our Method

Learning a transferable feature from source domain that can be used to improve performance on target domain has always been challenging. Finding causal relations between variables from the information theoretic view can be a plausible way to improve the transfer learning performance. While prior works [173] focus on finding invariance for better generalization ability, our method utilizes both domain-invariant features and domain-specific features to better serve our transfer learning goal.

We show the causal relationships between variables in Fig. 5.3. Follow [174], we decouple the randomness in the input variable X into two parts, one dependent on the task label Y and another random variable nuisance N independent of the label, and both label Y and nuisance N is dependent to the domain D. Inspired by [173],

we extract domain-invariant latent variable Z_i and domain-dependent latent variable Z_d from input X, then combine Z_i and Z_d to form a compressed latent variable Z_d^*.

According to the information bottleneck [175] and our causal graph Fig. 5.3, our learning objective is:

$$\underset{\theta_{g_i},\theta_{g_d},\theta_{f_{d^*}}}{\text{minimize}}\, I(Z_d^*, Y) - \lambda I(Y, D|Z_d^*) \tag{5.1}$$

Where g_i is the feature extractor (for domain invariant features), g_{dj} are adaptors for each domain, and f_{d^*} is the determinator. With the similar process of IIB [173], we can reform it as:

$$\underset{g_i,g_d,f_{d^*}}{\text{minimize}}\, \underset{f_i}{\text{minimize}}\, \mathcal{L}_{d^*}(g_i, g_d, f_{d^*}) + \lambda(\mathcal{L}_{d^*}(g_i, g_d, f_{d^*}) - \mathcal{L}_i(g_i, f_i)) \tag{5.2}$$

where

$$\mathcal{L}_{d^*} = \mathbf{E}_{x,y}[L(y, f_{d^*}(g_i(x), g_d(x)))] \tag{5.3}$$

$$\mathcal{L}_i = \mathbf{E}_{x,y}[L(y, f_i(g_i(x)))] \tag{5.4}$$

5.5.3 Experiments

Setups. All experiments are conducted using one Intel(R) Xeon(TM) W-2123 CPU, two Nvidia GTX 1080 GPUs, and 32G RAM. We use the Adam optimizer [68]

with learning rate 0.0001 and batch size 128 for training. The maximum number of epochs is 1,000. Other setup is explained in Sec. 5.3.

Comparison with other task adaptation methods. In Table 5.6, we compare our method with other SOTA task adaptation methods, i.e., DELTA [33], BSS [34], StochNorm [35]. All of them use both source and target data and labels. Our method outperforms others by up to 2.29%.

Comparison with other domain adaptation methods. In Table 5.6, we compare our method with other classic domain adaptation methods, i.e., DANN [176], ADDA [177], BSP [178]. Since those domain adaptation methods do not use target domain labels, our method can achieve up to 15.45% improvement.

Ablation study. To show each component (LoRA is the adapter, STB stands for style transferred branch, DP stands for dynamic probability for each domain, and IBL stands for information bottleneck loss) takes effect, we do an ablation study on each of them and show results in Table 5.7. Results show that removing any component will lead to a performance drop, which means each of them does contribute to the final performance.

5.6 Conclusion

In autonomous driving, applying learned knowledge in a known domain to an unknown domain is still one of the key challenges due to the variety of the driving scenarios in the real world (and virtual world). Domain-agnostic learning, or transfer learning, make it possible to achieve knowledge transfer between different

domains. In this work, we investigate how training data (in terms of image style and data amount), network architecture (Batch Norm layers, Adapters), and training paradigm (finetuning, partially finetuning, reinitialization) influence the domain-agnostic learning in the end-to-end steering task. Based on the analysis, we propose a novel domain-agnostic learning framework with (1) domain-specific adapters and shared modules to separate domain-specific information and task-specific information; (2) style-transferred branch to help split domain-specific information; (3) gradually increased ratio of target domain data in each epoch for better knowledge transfer from the source to the target domain.

Limitations and Future Works: (1) The style-transferred branch relies on the style transfer network like CycleGAN. we plan to merge the idea of CycleGAN into our framework in the future. (2) The distribution of the steering values are highly unbalanced. We need to explore whether there's a better training paradigm that can take advantage of this specific prior.

5.7 Appendix

5.7.1 Style Transfer between Real and Virtual Domain

We show sample images from the datasets R (Real photos in Nvidia dataset), (Real photos in virtual style generated by GAN), (Real photos in virtual style generated by color remapping) in the first row from left to right, and V (Virtual images in Udacity dataset), T_C (Virtual images in real style generated by GAN), (Virtual images in real style generated by color remapping) in the second row from

left to right.

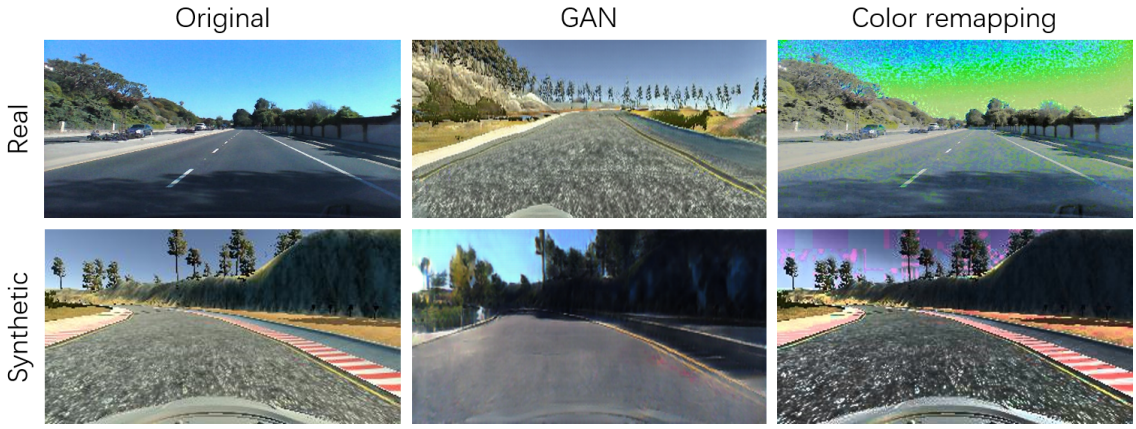


Figure 5.4: We show sample images from the datasets R (Real photos in Nvidia dataset), (Real photos in virtual style generated by GAN), (Real photos in virtual style generated by color remapping) in the first row from left to right, and V (Virtual images in Udacity dataset), T_C (Virtual images in real style generated by GAN), (Virtual images in real style generated by color remapping) in the second row from left to right.

5.7.2 Fréchet Inception Distance

We try to evaluate the domain gap with Fréchet Inception Distance (FID) [64], so that we don't need to train models when we met new domains. However, we found it's not necessary to be a proper metric when evaluating the domain gap for steering task. As shown in Table 5.8 (in Appendix 5.7.2), the relative order is different from the actual test results in Table 5.2.

	Model (M)	$\text{MA}_R(M)$
(a) Single dataset	train(R)	88.36%
	train($R1$)	32.02%
	train(V)	31.16%
	train(T_C)	26.87%
	train(T_M)	25.56%
(b) Simply combine	train($R1$ + R)	82.32%
	train(V + R)	75.74%
	train(T_C + R)	75.44%
	train(T_M + R)	76.85%
(c) Finetuning	train($R1$) → train(R)	81.93%
	train(V) → train(R)	83.54%
	train(T_C) → train(R)	82.70%
	train(T_M) → train(R)	79.04%
(d) Partially finetuning	train($R1$) → ptrain(R)	70.86%
	train(V) → ptrain(R)	73.66%
	train(T_C) → ptrain(R)	77.17%
	train(T_M) → ptrain(R)	72.97%
(e) Finetuning with reinitialization	train($R1$) → train(R)	**88.71%**
	train(V) → train(R)	**87.50%**
	train(T_C) → train(R)	**83.12%**
	train(T_M) → train(R)	**80.26%**
(f) Partially finetuning with reinitialization	train($R1$) → ptrain(R)	76.94%
	train(V) → ptrain(R)	75.08%
	train(T_C) → ptrain(R)	77.78%
	train(T_M) → ptrain(R)	74.28%

Table 5.3: Mean Accuracy comparison with different training paradigms. From (a) we can verify the existence of a domain gap between the virtual, style-transferred and real datasets. From (b,c,d,e,f), we find that (e)**"finetuning with reinitialization" outperforms other training paradigms**.

	M	$\text{MA}_R(M)$
(a) Finetuning + reinit header + BN	$\text{train}(V) \rightarrow \text{train}(R)$	80.53%
	$\text{train}(R1) \rightarrow \text{train}(R)$	80.77%
(b) AdvProp BN	$\text{train}(R, V)$	71.22%
	$\text{train}(R, R1)$	75.83%
(c) Finetuning + reinit header + LoRA	$\text{train}(V) \rightarrow \text{train}(R)$	**81.32%**
	$\text{train}(R1) \rightarrow \text{train}(R)$	**82.71%**

Table 5.4: Mean Accuracy comparison with different network architectures. **LoRA outperform others**.

	Model (M)	$\text{MA}_R(M)$
(a) Original network	train(R)	88.36%
	train($0.5R$)	80.23%
	train($0.1R$)	65.68%
	train($0.01R$)	55.41%
	train($0.001R$)	46.39%
(b) Original network, simply merge	train(R1 + R)	82.32%
	train(R1 + $0.5R$)	74.07%
	train(R1 + $0.1R$)	61.87%
	train(R1 + $0.01R$)	44.49%
	train(R1 + $0.001R$)	33.24%
(c) Original network, finetuning	train($R1$) $\rightarrow$ train(R)	81.93%
	train($R1$) $\rightarrow$ train($0.5R$)	79.94%
	train($R1$) $\rightarrow$ train($0.1R$)	65.00%
	train($R1$) $\rightarrow$ train($0.01R$)	53.06%
	train($R1$) $\rightarrow$ train($0.001R$)	38.83%
(d) network with BN layers	train(R)	81.33%
	train($0.5R$)	73.75%
	train($0.1R$)	62.64%
	train($0.01R$)	55.98%
	train($0.001R$)	47.97%
(e) network with BN layers, simply merge	train(R1 + R)	78.45%
	train(R1 + $0.5R$)	70.95%
	train(R1 + $0.1R$)	62.64%
	train(R1 + $0.01R$)	53.51%
	train(R1 + $0.001R$)	**48.24%**
(f) network with BN layers, finetuning	train($R1$) $\rightarrow$ train(R)	80.77%
	train($R1$) $\rightarrow$ train($0.5R$)	76.36%
	train($R1$) $\rightarrow$ train($0.1R$)	64.79%
	train($R1$) $\rightarrow$ train($0.01R$)	**58.92%**
	train($R1$) $\rightarrow$ train($0.001R$)	**53.09%**
(g) AdvProp BN	train($R1, R$)	75.83%
	train($R1, 0.5R$)	73.12%
	train($R1, 0.1R$)	55.53%
	train($R1, 0.01R$)	41.33%
	train($R1, 0.001R$)	41.13%

Table 5.5: Mean Accuracy of the experiments using different ratio of the original dataset R, and different experiments of adding R1 to R to improve the performance. Only the last two rows of part (f) is better than the baseline (last two rows in part (a)).

		$\text{MA}_R(M)$ (%) on different angle threshold τ (degree)						
	Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	**mAcc**	**MSE**
	Baseline	59.5%	82.1%	93.9%	96.3%	98.6%	86.04%	1.96
(a) Domain Adaptation	DANN [176]	28.9%	52.5%	79.3%	92.2%	97.3%	70.04%	0.58
	ADDA [177]	33.6%	54.3%	84.4%	93.2%	97.5%	72.6%	0.43
	BSP [178]	38.9%	60.4%	87.5%	95.1%	98.4%	76.06%	0.32
(b) Task Adaptation	DELTA [33]	61.9%	80.9%	93.9%	97.7%	99.2%	86.72%	0.16
	BSS [34]	67.0%	83.4%	93.8%	97.5%	98.8%	88.1%	0.21
	StochNorm [35]	53.7%	78.5%	92.8%	97.3%	99.2%	84.3%	0.18
	Ours	70.5%	84.3%	93.8%	97.9%	99.4%	**89.2%**	**0.15**

Table 5.6: Mean Accuracy comparison with domain adaptation and task adaptation methods. Our method outperforms others under all metrics.

	$\text{MA}_R(M)$ (%) on different angle threshold τ (degree)						
Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	**mAcc**	**MSE**
Baseline	59.5%	82.1%	93.9%	96.3%	98.6%	86.04%	1.96
Ours w/o LoRA	58.4%	80.3%	93.4%	97.7%	98.6%	85.68%	0.19
Ours w/o STB	68.0%	81.6%	94.1%	97.7%	99.0%	88.08%	0.16
Ours w/o DP	65.6%	82.2%	93.4%	97.3%	98.8%	87.46%	0.18
Ours w/o IBL	69.3%	84.0%	**93.9%**	97.5%	99.0%	88.74%	0.18
Ours	**70.5%**	**84.3%**	93.8%	**97.9%**	**99.4%**	**89.2%**	**0.15**

Table 5.7: Ablation study. LoRA is the adapter, STB stands for style transferred branch, DP stands for dynamic probability for each domain, and IBL stands for information bottleneck loss.

	R	V	RV_{CGAN}	VR_{CGAN}	RV_{CR}	VR_{CR}
R	0	200.62	226.50	146.03	40.00	265.36
V	200.62	0	117.25	198.12	173.29	107.64
RV_{CGAN}	226.50	117.25	0	152.52	177.01	168.51
VR_{CGAN}	146.03	198.12	152.52	0	119.46	222.60
RV_{CR}	40.00	173.29	177.01	119.46	0	230.15
VR_{CR}	265.36	107.64	168.51	222.60	230.15	0

Table 5.8: Fréchet Inception Distance between different datasets.

Chapter 6: Inverse Reinforcement Learning with Hybrid-weight Trust-region Optimization and Curriculum Learning for Autonomous Maneuvering

6.1 Introduction

Autonomous driving generally can be realized via either an end-to-end system or a mediated-perception approach [179]. The former takes in raw sensor data and directly output control commands (e.g., steering angles), which usually results in a succinct training pipeline at the cost of model interpretability. The later decouples perception and navigation, thus offering better model interpretability with enhanced driving safety. However, a mediated-perception approach commonly adopts a planning algorithm for navigating a self-driving car, which can be computationally expensive, given the requirement of holistic environment information for achieving global optimality and planning in high-dimensional state space.

We propose an efficient and novel mediated-perception framework for autonomous driving that exploits context-aware multi-sensor perception for inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning (IRL-HC). The perception module interprets unstructured information

(i.e., images and point clouds) of an environment using multiple sensors, extracts context-aware information, and produces structured information (e.g., the shape and position of an object). IRL-HC then takes the structured information along with expert trajectories as input to learn a control policy for autonomous vehicles (AV).

Our novel IRL-HC contains two main elements: (a) hybrid-weight trust-region optimization and (b) curriculum learning. Hybrid-weight trust-region optimization addresses a fundamental limitation of the original IRL [42]: a *uniform prior* is imposed on all features, which can lead to subpar imitation performance of the learner to the expert *even their feature expectations converge and match* [180]. In the context of autonomous driving, since some hard constraints like collision avoidance for safety and goal-oriented navigation to destination are most critical over other considerations, this limitation due to uniform priors in weight tuning can lead to frequent collisions. Our method alleviates this problem by imposing a *non-uniform prior* on task features and updating the features' weights using hybrid-weight *trust-region optimization*, which is progressively updated to automate weight-tuning optimization. This design of *hybrid-weight* optimization enables the use of both expert demonstrations and domain knowledge for learning an effective policy that takes into account of both experts (e.g., driver) and desired constraints (e.g., reaching the goal without collision). The second element of IRL-HC is curriculum learning, which has been shown effective in improving RL agents' performance [45]. Our key insight is that the trust-region is naturally linked to the estimation of the task learning progress—which can then be used to determine the difficulty of the task curriculum that is

built on the maximum step size for the learner to progress on the expert's trajectory.

In summary, we introduce a novel, efficient IRL framework consisting of hybrid-weight trust-region optimization and curriculum learning (IRL-HC), supported by a mediated-perception module that provides context-aware multi-sensor perception, as shown in Fig. 6.1. It offers several advantages:

- Hybrid-weight trust-region optimization improves upon IRL [42] by imposing non-uniform priors on task-critical features, e.g., collision avoidance and goal-seeking, incorporating *both* expert demonstration and domain knowledge to automate weight tuning effectively;
- Curriculum learning retains important lessons through *increasingly difficult* RL to task learning, thus improving overall training efficiency and performance;
- Curriculum learning also utilizes the hybrid-weight trust-region optimization to *assess curriculum difficulty*;
- IRL-HC is further compatible with domain-dependent techniques, such as learn-from-accident [48], which generate safe trajectories, further boosting the overall performance in autonomous driving.

The effectiveness and efficiency of IRL-HC are demonstrated in a variety of experiments. To show IRL-HC can work without perfect perception (i.e., use ground-truth data), we run IRL-HC in the autonomous system described in Sec. 6.3.1. Overall, our method can enable the vehicle to drive safely up to **10x** further than

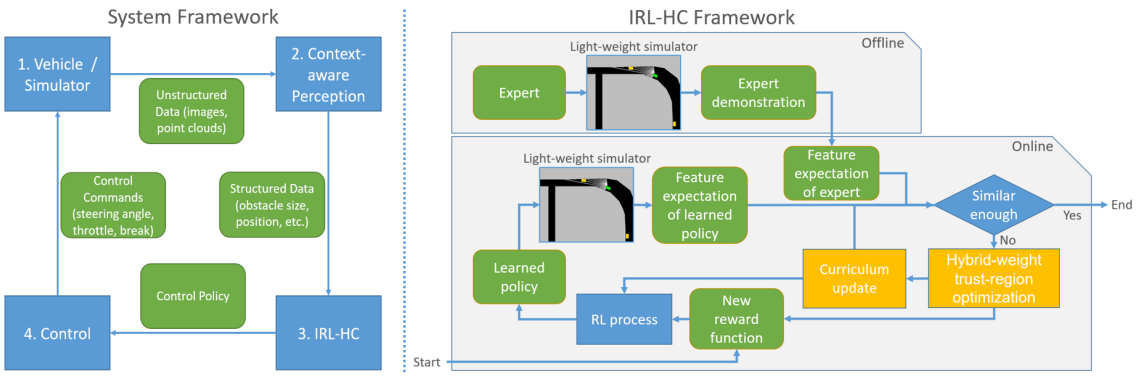


Figure 6.1: **System Pipeline (LEFT)**. At each time step, the vehicle/simulator generates unstructured data such as images and point clouds. These data are processed by the perception module to produce structured data, which are then used by the IRL-HC module to learn a control policy for autonomous driving. **IRL-HC training process (RIGHT).** We compute the expert trajectories offline. Next we use hybrid-weight trust-region optimization to obtain a new reward function, which is then used to compute the feature expectation of a learned policy online. The curriculum difficulty will also be updated progressively, according to the trust-region updating rules that assess the learned task difficulty.

existing SOTA methods, assist in reducing the number of collisions up to **48%**, and **2.8x** faster training. In addition to the statistical results, we show that IRL-HC can steer the vehicle to avoid both static and dynamic obstacles, even in the presence of a narrow passage (see project website).

6.2 Related Work

Various methods [28, 181] have been proposed to address the perception, planning, and control of an **autonomous vehicle** (**AV**). Examples of the end-to-end approach include end-to-end reinforcement learning [182] and end-to-end imitation learning [48, 183, 184, 185, 186]. These approaches usually require a large amount of training data in order to be robust in rare cases, such as pre-accident scenarios. In addition, the use of deep neural networks in these approaches to directly map

raw sensor data to control commands can lead to low model interpretability.

Examples of the mediated-perception approach include perception plus motion planning [187] and perception plus learning-based planning [183]. Because of the decomposition of perception and navigation, these approaches enjoy better model interpretability, hence improved driving safety. Recently, Li et al. propose ADAPS [48], an end-to-end imitation learning framework that enables an AV to learn from accidents. Compared to ADAPS, our work proposes a new architecture that can generalize better as it is based on RL rather than supervised learning for imitating the expert.

As an effective technique for imitation learning, IRL involves two steps: 1) learning a reward function from experts' demonstrations and 2) using the acquired reward function for RL to learn a control policy [42]. To provide some examples, Sharifzadeh et al. [43] apply Deep Q-Networks to extract a reward function in large state space. You et al. [44] use deep neural networks to approximate the latent reward function of the expert and then apply deep Q-learning to obtain the control policy.

Maximum entropy IRL [188] provides a probabilistic approach based on the entropy concept. Hierarchical guidance [189] leverages the hierarchical structure of the underlying problem to integrate different modes of expert interactions. Guided Cost Learning [190] uses a nonlinear cost function and guided sampling strategy. GAIL [191] draws the analogy between imitation learning and generative adversarial networks and develops an effective model-free imitation learning for complex, high-dimensional tasks. Maximum entropy deep IRL [192] combines neural networks and

traditional IRL. Another work [193] proposes a scalable IRL algorithm based on an adversarial reward learning process. Brown et al. [194] study the suboptimal demonstrations in IRL. Some recent studies [180, 195] explore the feature design in IRL to better meet human's intention. Compared to other IRL-based methods, our approach incorporates both expert demonstrations and domain knowledge to design an effective initial reward function for obtaining an effective policy via hybrid-weight trust-region optimization and curriculum learning. Our framework can also be used with feature design [180, 195] at ease.

Curriculum learning for RL has gained much attention recently [45, 46]. Early studies use curricula for tasks such as grammar learning [196] and robotics control problems [197]. Some well-known work such as AlphaGo [47] implicitly use curricula to guide training. Narvekar et al. [198] propose a meta-MDP to select tasks for the learning agent. Recently Song et al. [199] propose a three-stage curriculum RL to train an autonomous racing agent.

6.3 Approach

6.3.1 Framework Overview

Our framework combines context-aware multi-sensor perception and inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning (IRL-HC). The perception module takes multiple sensors input and produces structured data, which are then used by IRL-HC to learn a control policy, see Fig. 6.1 (LEFT).

We assume that the AV can obtain a global map from an external service, and compute its position and rotation with on-board GPS and Inertial Measurement Unit (IMU). We further assume that the AV knows beforehand a few sparse waypoints on its path to the goal, and the task of the AV is converted to reach the waypoints consecutively.

Our perception module can produce semantic-rich structured data to learn a reward function by utilizing the context-aware semantic information. Features with clear semantic interpretations can help construct non-linear features such as "whether there is a car in front within 3 meters", resulting in more flexible decision-making. This is particularly useful considering that IRL restricts the reward function to be a linear combination of the features.

The IRL-HC training process, shown on the right of Fig. 6.1, consists of an offline step and an online step. In the offline step, we use a planning algorithm as the expert to generate driving trajectories. In the online step, we learn the feature weights with hybrid-weight trust-region optimization given the expert's policy and the learned policy at the current iteration and non-uniform prior. Then, we construct a reward function using the learned feature weights. The difficulty of curriculum will be updated according to the trust-region updating rule. By further adopting the notion of *learn from accidents* [48], we use the resulting (additional) training data along with the newly constructed reward function for RL to update the learned policy. Inspired by transfer learning [200], the model parameters from the previous iteration are used as a starting point in continue training.

6.3.2 Context-aware Multi-sensor 3D Perception

We simulate three RGB cameras for the front-view, left-view, and right-view, respectively, as well as a 360-degree Lidar of the AV. We combine one RGB image and the point cloud on each side to detect nearby vehicles. Then, we merge the results from all sides to obtain an overall view. The module works as follows.

First, it generates the front-view data, which contain segmentation and depth maps, and bird's-eye-view image from the point cloud. By having the calibration data between the Lidar and camera, we can then align the front-view data with the RGB images. Next, we combine and feed the aligned front-view data with the RGB images into the feature extractor to obtain the front-view feature map. We apply the same procedure for the bird's-eye-view image to obtain the bird's-eye-view feature map. We use the region proposal network (RPN) [201] and detection network from AVOD [202] to obtain the perception results for the front-view. Similarly, we obtain the perception results for the left-view and right-view. Finally, we merge all perception results together using the extrinsic calibration data of the three cameras. The output of the perception module contains 3D bounding boxes and types of nearby dynamic obstacles. Our approach can also be extended to detect static obstacles when needed. The context-aware, multi-sensor 3D perception detects different types of information to produce features used by IRL.

6.3.3 IRL with Hybrid-weight Trust-region Optimization

We first review original IRL, point out some issues with IRL, then present our solution by extending the *trust-region optimization* to our scenario. To the best of knowledge, our method is the first one that can take advantages of both demo (expert data) and domain knowledge (reward function), leading to overall better performance.

The original IRL achieves imitation learning by first computing the expert's feature expectation $\hat{\mu}(\pi_E) = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t\phi(s_t^{(i)})$, given m trajectories $\{s_0^{(i)}, s_1^{(i)}, \dots\}_{i=1}^{m}$ from the expert's policy π_E, the discount factor γ, and the feature vector $\phi(\cdot)$. Then, IRL learns w ($w \in \mathbb{R}^k$ and $\|w\|_1 \leq 1$) given π at the current iteration and π_E, and synthesizes a reward function $R(s) = w \cdot \phi(s)$, where s is the state of the environment. Next, the policy π is re-learned using RL. This iterative process continues until $\|\mu(\pi) - \hat{\mu}(\pi_E)\|_2 \leq \epsilon$. The final policy is selected among all learned policies from all iterations.

The features used by IRL-HC are based on structured data from the perception module, which include bounding boxes of nearby vehicles and static obstacles in the scene. See detailed feature list in Sec. 6.4.

One fundamental limitation of IRL is that it imposes a *uniform prior* on all features, causing small weights to be possibly assigned to some crucial features during the learning process. For example, in the context of driving, we find that the feature *collision* can receive a small weight as a result of any collision behavior of the AV will terminate a training episode. This limitation of IRL can lead to subpar task

performance (see Sec. 6.4.3). To give an example, the expert can drive the car safely (without any collision), while a randomly initialized policy can hardly drive the car far without collision. In this case, the feature expectation of the expert is calculated using extended, "global" trajectories, while the feature expectation of the learned policy is calculated using short, "local" trajectories. This discrepancy is likely to cause some important features to receive small weights during the minimization process of the feature expectations in IRL. A concrete example of this phenomenon from our experiments is shown in Table 6.1.

feature	left dist.	front dist.	right dist.	...	**collision**
weight	0.113	0.184	-0.124	...	-0.00000534
$\mu(\pi_E)$	139.14	369.50	25.84	...	0
$\mu(\pi_0)$	40.31	216.06	142.87	...	0.005

Table 6.1: **Limitations of IRL.** IRL learns feature weights w by minimizing the difference of feature expectation between the expert's policy $\mu(\pi_E)$ and a random policy $\mu(\pi_0)$. While both policies have small expectations for the feature *collision*, the actual trajectory from the expert's policy can be much longer than the trajectory from a random policy. As a result, IRL assigns a negligible weight to *collision* (i.e., $-5.34e-06$).

To alleviate the aforementioned limitation of IRL, we propose an approach that not only incorporates a non-uniform prior on the features by allowing users to specify the weights of certain features for ensuring essential properties of a task, e.g., collision for driving, but also uses a hybrid weight tuning to update the policy so that the vehicle can adapt to different environments while maintaining desired behaviors. Formally, the overall weights $w = [w_m, w_l]$ consist of empirically-initialized weights $w_m = [w_1, \ldots, w_k]$ and the weights to be learned from scratch $w_l = [w_{k+1}, w_{k+2}, \ldots, w_n]$, where n is the total number of features. To achieve an

optimal policy in diverse environments, we use trust-region optimization [203] to automatically tune w_m by avoiding violent exploratory behaviors. Specifically, in the ith iteration, we first try to update ϵ and w by solving the following quasi-convex optimization program:

$$\epsilon^{(i)} = \max_{\substack{w^{(i+1)}:\|w^{(i+1)}\|_2\leq 1 \\ \|w_m^{(i+1)}-w_m^{(i)}\|_2\leq\Delta}} \{\min_{j\in\{0,\ldots,i\}} (w^{(i+1)})^T(\mu(\pi_E)-\mu(\pi^{(j)}))\}, \tag{6.1}$$

where Δ is the trust-region radius ($\Delta = \Delta^{(i)}$). The ratio $\epsilon^{(i)}/\epsilon^{(i-1)}$ is used to determine the acceptance of the newly updated w. Since the predicted upper bound of $\epsilon^{(i)}/\epsilon^{(i-1)}$ is $\frac{n}{\sqrt{n^2+(1-\gamma)^2\epsilon^2}}$ [42], we set the acceptance condition to

$$\frac{\epsilon^{(i)}}{\epsilon^{(i-1)}} \leq \alpha \frac{n}{\sqrt{n^2+(1-\gamma)^2\epsilon^{*2}}}, \tag{6.2}$$

where $\alpha < 1$ is the threshold parameter. If the condition is failed, we drop the newly found w, and update ϵ and w by solving Eq. 6.1 with $\Delta = 0$. As the last step of one iteration, we update Δ as follows:

$$\Delta^{(i+1)} = \begin{cases} \min(c_u\Delta^{(i)}, b_u) & \text{Eq. 6.2 met} \\ \max(c_l\Delta^{(i)}, b_l) & \text{Eq. 6.2 failed p times} \\ \Delta^{(i)} & \text{otherwise} \end{cases} \tag{6.3}$$

where b_u and b_l are the upper- and lower-bound of the trust-region radius, and $c_u > 1$ and $c_l < 1$ are the coefficients.

Additionally, IRL retrains π at each iteration, such a process can be inefficient. In IRL-HC, we improve the training efficiency by using the learned model parameters

Algorithm 4: Inverse Reinforcement Learning with Hybrid-weight Trust-region Optimization and Curriculum Learning (IRL-HC)

Input: expert trajectories
Output: policy $\pi^{(i)}$

Initialisation:
Calculate $\mu(\pi_E)$ with expert trajectories
Set $i = 0$, set $\epsilon, \gamma, \alpha, b_u, b_l, c_u, c_l, p, e_u, N_u$
Randomly set the model parameters $\theta^{(0)}$ for $\pi^{(0)}$
Compute $\mu(\pi^{(0)})$
Set $w_m^{(0)}$ such that $\|w_m^{(0)}\|_2 < 1$ (initial reward weights), $w_l^{(0)} = \mathbf{0}$
Compute $\epsilon^{(0)} = (w^{(0)})^T(\mu(\pi_E) - \mu(\pi^{(0)}))$, where $w^{(0)} = [w_m^{(0)} w_l^{(0)}]$
Set $\Delta^{(0)}, n_s^{(0)}$
while $\epsilon^{(i)} > \epsilon$ **do**
  Set $i = i + 1$
  Compute the reward function $R = ((w^{(i-1)})^T \phi)$
  Using R, $\theta^{(i-1)}$, and $n_s^{(i-1)}$ in RL to compute an optimal policy $\pi^{(i)}$
  Compute $\mu(\pi^{(i)})$
  Solve Optimization 6.1 with $\Delta = \Delta^{(i-1)}$, and get solution $\epsilon^{(i)}$ at $w^{(i)}$
  if Eq. 6.2 is True **then**
    Accept $\epsilon^{(i)}$ and $w^{(i)}$
  else
    Reject, solve Eq. 6.1 with $\Delta = 0$, and update $\epsilon^{(i)}$ and $w^{(i)}$
  Set $\Delta^{(i)}$ with Eq. 6.3 (trust-region update)
  Set $n_s^{(i)}$ with Eq. 6.5 (curriculum difficulty update)
=0

$\theta^{(i)}$ of $\pi^{(i)}$ at the ith iteration as the initial parameters for training $\pi^{(i+1)}$ at the $(i+1)$th iteration. The whole learning process stops when $\|\mu(\pi) - \hat{\mu}(\pi_E)\|_2 < \epsilon$.

In many cases, we can design a reward function to incorporate domain knowledge. For example, in autonomous driving we want the vehicle to reach the goal without collision; an example reward function could be $r = w_1 g - w_2 c$, where g is the "goal-reaching" flag, c is the collision flag, and w_1 and w_2 are positive coefficients. In complex scenarios such as our city environment, however, such a naive reward function does not work well: sometimes learned policy will let the car stop there forever to avoid collision, or run in a circle in a very small area (keep running with-

out collision but cannot achieve the goal). In this case, we can learn from expert demonstrations to complete the task.

We can represent any reward function as a sum of its linear/nonlinear components. To set the initial reward function for IRL-HC, we add each component of the reward function into the feature vector $\phi(\cdot)$ and add the corresponding coefficient into empirically-initialized weights w_m. For these components, since they already contain prior information, we do not want to aggressively update them (in fact in the beginning of IRL, aggressively updating feature components can deteriorate the learning performance because the initial policy is unstable)—this is achieved via hybrid-weight trust-region optimization.

Key Insight: Our method is different from *manually setting initial weights* for IRL. Notice the IRL update the feature weights by solving an optimization without the previous feature weights (although the previous feature weights will influence the feature expectation of the learned policy in the optimization implicitly, the uncertainty of RL process will reduce such relation). This means the updated weights can be quite different from the previous weights. *Adding the trust-region mechanism can bound the weights during the entire IRL process while allowing tuning within a safe region.* In addition, our method is different from simple *re-weighting mechanism* since ours contains an optimization under different constraints for two sets of parameters.

6.3.4 Curriculum Learning with Trust-region

Another problem of IRL is efficiency, since IRL contains multiple RL training process. We use Sequence Curriculum [45] in IRL-HC to help train IRL, aiming to improve the performance as well as the efficiency. Observing that the trust-region implicitly reflects the current learning progress, we use the trust-region updating criteria (Eq. 6.2) to determine when to increase the curriculum difficulty. Specifically, we use the maximum step number of each trajectory n_s as an curriculum difficulty variable (large n_s leads to an increase of the state space, thus increasing the learning difficulty). Meanwhile, we also monitor the safe step number of learned policy at each iteration to show the effectiveness of the policy learned so far:

$$n_{lp} \geq \beta \, \min(n_s^{(i)}, n_E) \tag{6.4}$$

where $\beta < 1$ is the threshold parameter, n_{lp} is the safe step number of the trajectory generated by the learned policy, n_E is the safe step number of the trajectory generated by the expert. n_s will be initialized with a small number and then increased with the following formula after each round of hybrid-weight trust-region optimization:

$$n_s^{(i+1)} = \begin{cases} \min(e_u n_s^{(i)}, N_u) & \text{Eq. 6.2 and Eq. 6.4 met} \\ n_s^{(i)} & \text{otherwise} \end{cases} \tag{6.5}$$

where $e_u > 1$ is the coefficient and N_u is the upper-bound of the n_s. Notice the curriculum is designed to be gradually harder, so n_s will not decrease, which is different than the trust-region radius. After the update of n_s, the RL process will use n_s and the new reward function given by hybrid-weight trust-region optimization to update the control policy. The IRL-HC algorithm is shown in Algorithm 4. In addition, *as the task difficulty gradually increases, we reuse learned parameters in continue training. This approach enables the transfer learning between tasks in curriculum learning and improves the performance of the learning agent.*

6.3.5 Learn from Accidents

IRL-HC is also compatible with other domain-dependent techniques. Inspired by ADAPS [48], we adopt *learn from accident* to improve the training efficiency. Specifically, when the car crashes during the online training process, we will backtrack for certain frames and let the expert drive for certain frames to avoid collision. Both the crash and collision-free data will be (re-)used to train the policy. Essentially, by imposing a non-uniform prior on extracted features, we can introduce certain expert experience to the learning process directly. Lastly, the process of *learn from accident* can result in both negative examples and positive examples in learning, leading to better performance and faster convergence.

6.3.6 Framework Design

We have introduced the key components of our framework and here we provide an integrated interpretation as well as module relation (see Fig. 6.2). Context-aware multi-sensor perception converts raw sensor data to structured information, providing flexibility in choosing features to be used in the hybrid-weight trust-region optimization. Next, *domain experience (via non-uniform prior) is incorporated by the optimization program into IRL for improving learning performance, and provides difficulty measurements for curriculum development. Curriculum learning then generates tasks with increasing difficulty to further accelerate the convergence rate of the learning agent.* Lastly, learn from accident provides data for corner cases, balancing the overall training data distribution.

6.4 Experiments and Results

In this section, we detail our experiments and results. All experiments are conducted using Intel(R) Xeon(TM) W-2123 CPU, Nvidia GTX 1080 GPU, and 32G RAM.

6.4.1 Overall Performance

We compare IRL-HC to the original IRL, Generative Adversarial Imitation Learning (GAIL) [191], adversarial inverse reinforcement learning (AIRL) [193], and end-to-end imitation learning (IM) [58]. We use deep Q-learning with [164,150] hidden units in each of the 2 dense layers and 20% dropout rate in all RL-based methods

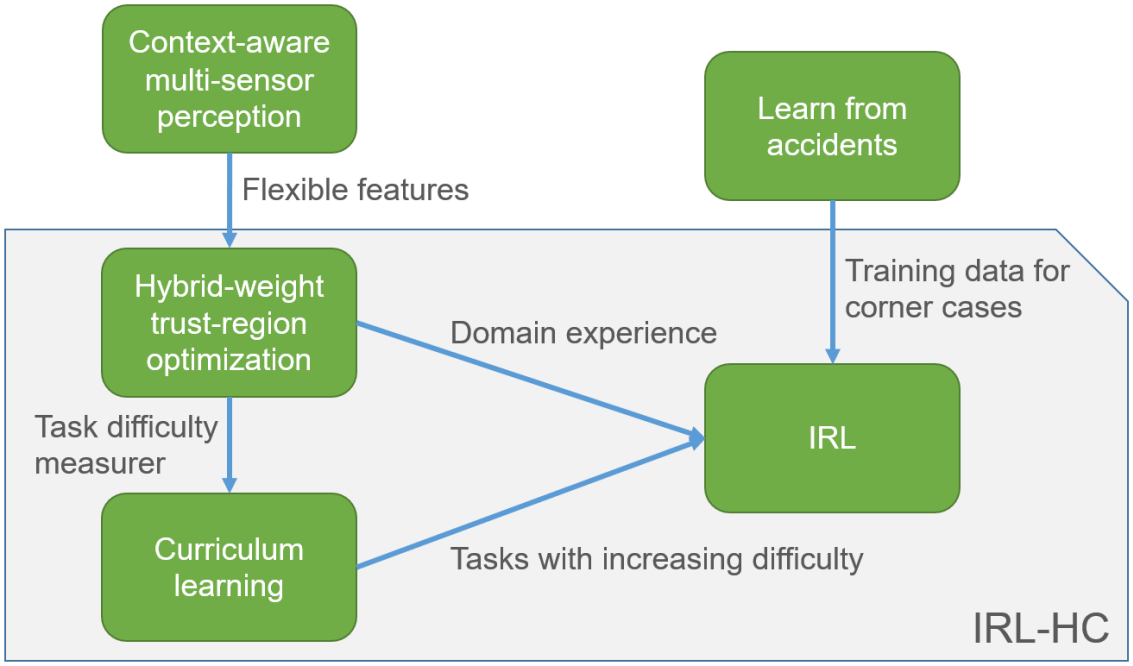


Figure 6.2: **Module relation of our framework.** The perception module converts sensor data to various features, which serve as input to the hybrid-weight trust-region optimization. Subsequently, curriculum learning is adopted to generate tasks with increased difficulty to train the learning agent. The learn from accident process further provides training data of corner cases to balance the data distribution, leading to overall better performance.

except for GAIL, where we use TRPO [204] as proposed in the original paper. We also experiment with Q-learning using more complex network architectures up to 6 hidden layers, but found [164,150] with 2 hidden layers works the best. The features used in IRL-based methods are listed in Fig. 6.3.

The action space contains 25 discrete {rotational speed, acceleration} action pairs using 5 levels of rotational speed (steering angle) and 5 levels of acceleration (throttle value and break value). We set policy similarity threshold $\epsilon = 0.1$ and discount factor $\gamma = 0.99$ for both IRL and IRL-HC. We collect 20,000 steps of the expert trajectory and train all methods for 24 hours (we also observe that training 12 more hours will not improve the performance). For IRL-HC, we set the trust-region

ID	Feature name	Description
1	Waypoint reaching flag	1 if a waypoint is reached 0 otherwise
2	Distance reduction to waypoint	Distance (meter) reduction to waypoint compared to the previous step
3	Direction away from waypoint	Relative angle (radian) between the forward direction of AV and direction to waypoint
4	Collision flag	1 if collide 0 otherwise
5~25	Depth list	Depth (meter) in 21 directions uniformly distributed in 120 degree ([-60, 60]) of the forward direction in bird's-eye-view
26~46	Object type list	Object type of the first object that stops the ray from AV, in 21 directions uniformly distributed in 120 degree ([-60, 60]) of the forward direction in bird's-eye-view 0 if nothing is detected or unknown object 1 if static object 2 if dynamic object

Figure 6.3: **Features used in IRL and IRL-HC (46 in total).**

acceptance coefficient $\alpha = 0.9$, trust-region increasing and decreasing coefficients $c_u = 1.1, c_l = 0.9$, upper- and lower-bound $b_u = 0.05, b_l = 0.001$, and consecutive rejection number $p = 5$. We empirically initialize weights for the first 4 features in Fig. 6.3 with values [0.4, 0.01, -0.1, -0.8] (represents the initial reward function), and let IRL-HC learn the weights of depth- and object type-related features. The perception network is trained using 10,000 frames of simulated data.

Our evaluation criterion is *the distance travelled by the AV under a fixed number of steps without any collision*. The AV will stop if it finishes $1,000$ steps or is in collision. Since the AV is assumed to know beforehand a series of waypoints on its path to the goal, we compute the score based on the number of waypoints reached

by the AV:

$$s_{final} = (n_{reached} + (1 - \frac{dist_{next}}{dist_{last\_next}})) \times s_{unit}, \tag{6.6}$$

where s_{final} is the final score, $n_{reached}$ is the number of waypoints reached, $dist_{next}$ is the distance between the last position on the car's trajectory and the next waypoint on the car's route, $dist_{last\_next}$ is the distance between the last reached waypoint to the next waypoint, and s_{unit} is the unit score by reaching a waypoint, which is set to 100. Note that a negative score may appear if $dist_{next} > dist_{last\_next}$, which happens when the car drives away from the next waypoint. We use three scenes shown in Fig. 6.4 for evaluation (test field about 160m x 160m):

- Scene 1: open space with random moving vehicles;
- Scene 2: city street with static obstacles;
- Scene 3: city street with random moving vehicles and static obstacles.

Figure 6.4: Screenshots of the three scenes used in our evaluation. From left to right: Scene 1, Scene 2, Scene 3.

6.4.2 Comparison with Other SOTA Methods

We record the final scores $s_{final,2}$ and $s_{final,3}$ from Scene 2 and 3; and average trajectory length $l_{final,1}$, $l_{final,2}$, $l_{final,3}$ from Scene 1, 2, and 3. Because s_{final} is

computed based on the waypoint position and there is no explicit waypoint in Scene 1, we do not compute $s_{final,1}$. All data are obtained by learning 3 times and testing 100 times with randomly initialized scene configurations such as the start position and direction of the AV, and the start position, direction and speed of obstacle vehicles. Our method achieves the highest scores and can enable the AV to drive safely 10x further (and longer) than the other methods. The full results are shown in Table 6.2.

Method	$s_{final,2}$	$s_{final,3}$	$l_{final,1}$	$l_{final,2}$	$l_{final,3}$
IM [58]	77.4	60.1	105.6 m	53.7 m	44.7 m
IRL [42]	110.7	59.7	228.8 m	69.4 m	33.2 m
GAIL [191]	103.0	52.8	49.1 m	69.9 m	35.1 m
AIRL [193]	119.9	83.6	74.1 m	73.6 m	50.7 m
Ours	**203.2**	**179.6**	**279.1** m	**733.5** m	**335.9** m

Table 6.2: **Performance of IRL-HC vs. other SOTA methods**, using the score defined in Eq. 6.6 and safe-trajectory length. Our method not only achieves the highest scores but also enables the AV to drive safely 10x further than the other methods.

We also compare performance between methods using reward functions vs. expert demonstration in Table 6.3. The reward function for RL is the same as the initial reward function for our method. IRL with only expert data cannot learn well either, while our method with both expert data and reward functions performs the best.

The analysis of the results shown in Table 6.2 and Table 6.3 is the following. RL relies on a good reward function constructed using domain knowledge, which can be difficult to obtain when addressing complex control tasks. IM does not generalize well to new environments since it "copies" the expert trajectory. IRL, GAIL, or

Method	$s_{final,2}$	$s_{final,3}$	$l_{final,1}$	$l_{final,2}$	$l_{final,3}$
RL (R)	99.8	59.1	72.9 m	59.7 m	39.7 m
IRL (E)	110.7	59.7	228.8 m	69.4 m	33.2 m
Ours (E+R)	**203.2**	**179.6**	**279.1** m	**733.5** m	**335.9** m

Table 6.3: **Overall performance comparison between methods using reward function vs. expert data.** RL with reward function (R) or IRL with only expert data (E) does not perform well on the task, while ours with both (E+R) performs the best.

AIRL only considers the expert trajectory while our method is able to combine expert trajectory and domain knowledge for learning. The hybrid-weight trust-region optimization enables our method to adapt to different environments. The curriculum learning further enables the learning agent to behave in gradually more difficult tasks while the reuse of the model parameters helps improve training efficiency. In addition, learn from accidents provides the RL agent with more training data for corner cases. Each above-mentioned element has contributed to the performance boost of the learning agent, thereby resulting in considerably better overall performance than other methods shown in Table 6.2. Note that all these analyses are not restricted to the autonomous driving task but likely to be relevant for other robot learning tasks.

6.4.3 Ablation Study

Here, we first show the effectiveness of individual components of our framework in Table 6.4. H, HC, LfA stand for "hybrid-weight trust-region optimization," "hybrid-weight trust-region optimization and curriculum learning," and "learn from accidents," respectively. Then, we show the combination of all components leads to

the best performance. Note that the experiment of "IRL + curriculum learning" is omitted since the curriculum design depends on the trust-region optimization.

Method	$s_{final,2}$	$s_{final,3}$	$l_{final,1}$	$l_{final,2}$	$l_{final,3}$
IRL [42]	110.7	59.7	228.8 m	69.4 m	33.2 m
IRL + H	130.7	103.4	235.6 m	140.4 m	133.9 m
IRL + HC	174.9	137.4	253.8 m	305.6 m	202.8 m
IRL + LfA	124.8	84.6	241.4 m	103.6 m	61.5 m
Ours (IRL+ALL)	**203.2**	**179.6**	**279.1** m	**733.5** m	**335.9** m

Table 6.4: **Ablation study of individual components.** The combination of all components leads to the best overall performance.

Next, we show the effectiveness of the main attributes, i.e., hybrid-weight trust-region optimization and curriculum learning, using the number of collisions encountered by the vehicle. To be specific, we count the number of collisions from the original IRL, IRL+H, and IRL+HC by running all approaches for 10,000 steps. As shown in Table 6.5, IRL+HC can reduce the number of collisions up to 48%.

Model	**Scene 1**	**Scene 2**	**Scene 3**
IRL	35	58	111
IRL+H	33	41	93
IRL+HC	**25**	**30**	**78**

Table 6.5: **The number of collisions in different scenes** over 10,000 steps: original IRL vs. IRL+H vs. IRL+HC. IRL+HC (our approach) can reduce the number of collisions up to 48%.

Except for performance, curriculum learning in our approach also aims to improve training efficiency. From the results shown in Fig. 6.5, we can see that by having this attribute, we can achieve comparable model performance at *2.8x faster*.

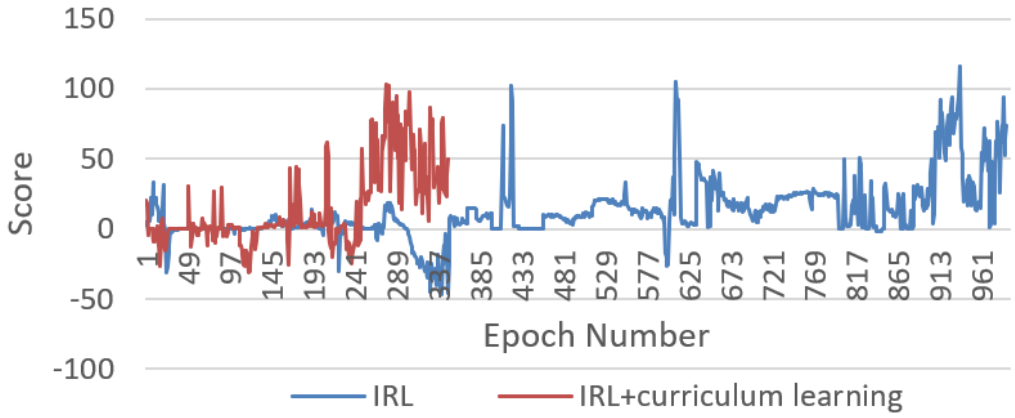


Figure 6.5: Training efficiency comparison. Using curriculum learning, we can achieve comparable score (model performance) **2.8x** faster.

6.4.4 Demo Driving Cases

Our method can enable safe autonomous driving by avoiding both static and dynamic obstacles. We show the results of some driving cases achieved by our method in Fig. 6.6. The full demo video can be found on the project website.

In Fig. 6.6(a), we show that our method can steer the AV to make a left turn around the static obstacle while maintaining safe driving; In Fig. 6.6(b), we show that our method can lead the AV to avoid both static and dynamic obstacles. In particular, the AV (in green) steers to the right to avoid another vehicle coming from the opposite direction while moving away from the static obstacle; Lastly, we show that our approach can avoid multiple dynamic obstacles in Fig. 6.6(c), where the AV (in green) is able to make a left turn to pass a narrow space between two other vehicles.

6.5 Conclusion and Future Work

We propose a framework that utilizes context-aware multi-sensor perception to enable inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning (IRL-HC) for autonomous driving. The hybrid-weight trust-region optimization empowers the network to incorporate both expert demonstrations and domain knowledge into learning. The curriculum learning enables the vehicle to perform well gradually on a series of increasingly difficult tasks. The process of learn from accidents further provides our learning agent with training data on edge cases to improve its task performance. We evaluate our approach using a variety of experiments, over the entire algorithm, and perform ablation study on the contribution of each individual component. As shown in all comparisons, our method outperforms the state-of-the-art methods on all measures.

Limitations and Future Work: First, we observe that the training results are relatively sensitive to the initial conditions. Second, our approach inherits the limitations of IRL, for example, information loss as a result of encoding expert trajectories into a single feature expectation. In the future, we hope to alleviate the information loss issue. Finally, we plan to test our approach in dense virtual traffic [205] reconstructed using real-world traffic data [206, 207, 208].

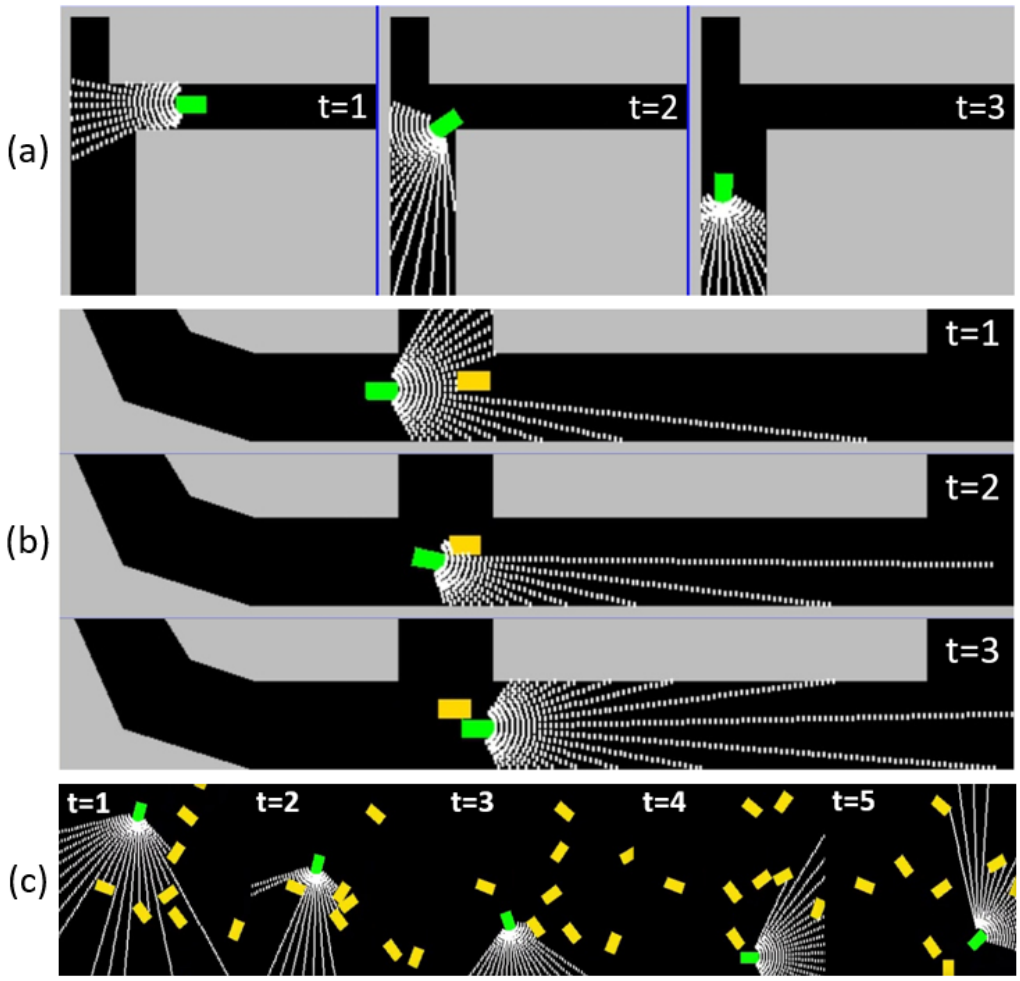


Figure 6.6: Driving case analysis. (a) **Static obstacle avoidance.** Our method can lead the AV to make a left turn to avoid a static obstacle while maintaining safe driving. (b) **Static and dynamic obstacle avoidance.** The AV (in green) can avoid another vehicle (in yellow) coming from the opposite direction by steering away from the static obstacle. (c) **Collision avoidance with multiple dynamic obstacles.** Our method can direct the AV (in green) to avoid all nearby vehicles even when a narrow passage is presented.

Chapter 7: Conclusion

In this dissertation, I proposed various methods for improving the robustness, generalizability, and performance of autonomous driving from data in the front-end and policy in the back-end. These methods enable (1) robust autonomous driving by sensitivity analysis and adversarial training, (2) cost-effective autonomous driving by distilling knowledge from multi-modality to single-modality, (3) generalized autonomous driving by employing transfer learning from virtual to real and from real to real environments, and (4) better policy learning in autonomous driving through inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning. The effectiveness and accuracy of these approaches have been demonstrated through extensive experimentations. Overall, the proposed works greatly boost the robustness and generalizability of autonomous driving.

7.1 Summary of Results

Chapter 2 first analyzes the influences of different image-quality attributes. By using Fréchet Inception Distance (FID) as a unified metric, it conducts sensitivity analysis using Mean Accuracy (MA) vs. FID, or in the MA-FID space. Leveraging the insights gained from the sensitivity analysis, an effective and efficient training

method to enhance the generalization of learning-based steering under various image perturbations has been proposed. This method can be readily extended and applied beyond the set of factors and learning algorithms investigated in this study, as well as other multimodal sensor data and tasks. In addition to the significant improvement in the task performance on the base datasets, performance in datasets with a mixture of perturbations and previously unseen adversarial examples has also been remarkably enhanced.

Chapter 3 introduces '*Auxiliary Modality Learning (AML)*'. It first formalizes the concept of AML in terms of types of auxiliary modality and architectures for AML, then analyzes how types of auxiliary modality and architectures can affect AML performance *on a single task* and *across tasks*. Also, the effectiveness of AML from optimization and data perspectives has been assessed to provide theoretical support for AML. Based on the aforementioned findings, we propose a novel method, Smart Auxiliary Modality Distillation (SAMD), to first determine the most suitable auxiliary modality, and then employ a special auxiliary modality distillation to enable the teacher network to be aware of the student's status, leading to a better distillation that achieves the SOTA performance.

Chapter 4 studies the problem of how to introduce varying amounts of auxiliary modality data to boost the performance of single modality learning in an end-to-end steering task. It proposes a new framework, **AMD-S-Net**, which can take in the main modality and varying amounts of auxiliary modality data to address this problem. In addition, a novel training paradigm that utilizes *reset* operation to facilitate knowledge transfer has been proposed. The AMD-S-Net and training

paradigm can offer superior performance compared to other SOTA methods in the field.

In autonomous driving, applying learned knowledge from a known domain to an unknown one is still one of the key challenges due to the variety of driving scenarios in both real and virtual environments. Domain-agnostic learning, or transfer learning, makes it possible to achieve knowledge transfer between different domains. Chapter 5 investigates how training data (in terms of image style and data amount), network architecture (Batch Norm layers, Adapters), and training paradigms (finetuning, partially finetuning, reinitialization) influence domain-agnostic learning in the end-to-end steering task. Based on the analysis, we propose a novel domain-agnostic learning framework with (1) domain-specific adapters and shared modules to separate domain-specific and task-specific information; (2) style-transferred branches to help segregate domain-specific information; (3) a gradually increasing ratio of target domain data in each epoch for better knowledge transfer from the source to the target domain.

Finally, Chapter 6 proposes a framework that utilizes context-aware multi-sensor perception to enable inverse reinforcement learning with hybrid-weight trust-region optimization and curriculum learning for autonomous driving. The hybrid-weight trust-region optimization empowers the network to incorporate both expert demonstrations and domain knowledge into learning. The curriculum learning enables the vehicle to perform well gradually on a series of increasingly challenging tasks. The process of learning from accidents further provides our learning agent with training data on edge cases, thereby enhancing its task performance. The

proposed method has been evaluated using a variety of experiments over the entire algorithm, and an ablation study has been performed on the contribution of each individual component. As shown in all comparisons, our method consistently outperforms the state-of-the-art methods on all measures.

7.2 Limitations

While the proposed methods have demonstrated effectiveness, they have four key inherent limitations.

First, the **driving domain prior is not fully utilized**. A common limitation of learning-based methods is the lack of explanation. If domain knowledge is applied to the learning models, the system could be more explainable. Given the specific nature of autonomous driving, each driving task has its specific priors. For instance, in the end-to-end steering task, the label distribution from a natural driving trajectory will be extremely imbalanced (with the steering angle close to zero in most cases). The dataset and data distribution are manually made more balanced in our work during the pre-processing, implying that a significant amount of data with near zero steering angle is dropped, in which valuable driving environment information is contained. This pre-processing may potentially impede performance improvement. Therefore, finding effective ways to utilize such priors is an area to be investigated

Second, the **network architecture could be optimized**. The proposed methods are mainly focused on the framework level. While users can take advantage

of the flexibility of changing the backbone, better network architecture designs may exist for a specific task in autonomous driving. In addition, in the works about auxiliary modality learning, the teacher model should be a "supermodel" of the student. While performance benefits from the supermodel condition can be achieved, the flexibility of model choice would be lost.

Another limitation pertains to the desire for **higher training efficiency**. My work mainly explores additional data from different perspectives to boost performance. However, generally speaking, more data will lead to more computation, e.g., adversarial training with data augmentation, auxiliary modality learning with data from other modalities, and transfer learning with data from other domains, will all lead to more training time compared to vanilla training. Furthermore, in the case of policy learning, the training efficiency problem is naturally inherited from the original inverse reinforcement learning, which contains multiple rounds of reinforcement learning. Balancing the trade-off between benefits from additional data and the associated time costs remains an open challenge.

Finally, my work has not been **implemented on a real-world autonomous vehicle**. It's challenging to obtain an autonomous vehicle for testing, due to logistical issues like financial implications or legislative constraints. Nevertheless, in order to build a powerful autonomous driving system, it is necessary to test algorithms on a real automobile, as both datasets and simulators are too ideal compared to the real-world environment, in which multiple factors like road conditions, hardware conditions, and other drivers' behavior are involved.

7.3 Future Work

Based on the limitations discussed in Section 7.2, there are several possible future directions.

Driving domain priors. Further research can be conducted to utilize the driving domain priors to enhance the explainability and performance of the learning-based autonomous system. These may include exploring specific training paradigms, network architectures, and loss functions – all of which are tailored to address the label distribution prior in the steering task. Additionally, the integration of driving knowledge to improve robustness, such as distinguishing perturbations on the road from those on roadside buildings, can be investigated. Furthermore, studying the selective sharing and distillation of information between modalities and domains can help establish a better autonomous system.

Network architecture. Existing common network architectures are usually designed for classic computer vision tasks initially, e.g., classification and segmentation. However, end-to-end driving involves both perception and control. While classic networks can be used in this kind of integrated tasks, it remains an open question whether there is a more efficient network that can combine vehicle dynamics and perception. Moreover, in my proposed auxiliary modality learning methods, the teacher model should be a "supermodel" of the student so that it can be aware of the student's status, a condition very strict for general knowledge distillation. Further exploration could focus on whether there are less constrained conditions

capable of producing a similar effect or other architectural conditions capable of achieving better performance.

Training efficiency. Multiple topics may be explored on improving training efficiency. From the perspective of input data, it can be investigated whether it is possible to reduce the size of input data, such as lowering image resolution, while maintaining comparable performance. From the network architecture perspective, the search for a more compact network architecture with fewer parameters but similar performance holds promise. From the training paradigm perspective, a more efficient way of training the network, such as dynamically selecting a training batch with the greatest performance improvement, can be studied. Lastly and from the optimization perspective, a better optimizer that accelerates model convergence is an avenue for further study.

Real-world autonomous vehicle. A direct solution for addressing this problem is to pursue collaboration or employment opportunities in autonomous driving companies. However, this solution may not be accessible to all students. An alternative is to conduct research on more advanced simulators with the latest AIGC technologies and more comprehensive physical-based models, or on robots with similar dynamic systems to cars.

Summary. To conclude, my future work will contain explorations on driving domain priors, network architecture, training efficiency, and real-world autonomous vehicle, to further extend my current work and make future autonomous driving systems more robust, efficient, safe, and affordable.

Bibliography

[1] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[2] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[3] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*, 2019.

[4] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *CVPR*, 2020.

[5] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *Advances in neural information processing systems*, pages 2797–2806, 2017.

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[7] S. Ghosh, R. Shet, P. Amon, A. Hutter, and A. Kaup. Robustness of deep convolutional neural networks for image degradations. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2916–2920, 2018.

[8] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[9] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020.

[10] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[11] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[12] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019.

[13] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 1147–1158, 2020.

[14] Yi Xiao, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, and Antonio M López. Multimodal end-to-end autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[15] Zhiyu Huang, Chen Lv, Yang Xing, and Jingda Wu. Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding. *IEEE Sensors Journal*, 21(10):11781–11790, 2020.

[16] Jyri Maanpää, Josef Taher, Petri Manninen, Leo Pakola, Iaroslav Melekhov, and Juha Hyyppä. Multimodal end-to-end learning for autonomous steering in adverse road and weather conditions. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 699–706. IEEE, 2021.

[17] Judy Hoffman, Saurabh Gupta, and Trevor Darrell. Learning with side information through modality hallucination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 826–834, 2016.

[18] Nuno C Garcia, Pietro Morerio, and Vittorio Murino. Learning with privileged information via adversarial discriminative modality distillation. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2581–2593, 2019.

[19] Nathan Piasco, Désiré Sidibé, Valérie Gouet-Brunet, and Cédric Demonceaux. Improving image description with auxiliary modality for visual localization in challenging conditions. *International Journal of Computer Vision*, 129(1):185–202, 2021.

[20] Lan Wang, Chenqiang Gao, Luyu Yang, Yue Zhao, Wangmeng Zuo, and Deyu Meng. Pm-gans: Discriminative representation learning for action recognition using partial-modalities. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 384–401, 2018.

[21] Nuno C Garcia, Pietro Morerio, and Vittorio Murino. Modality distillation with multiple stream networks for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[23] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[24] Sully Chen. A collection of labeled car driving datasets, https://github.com/sullychen/driving-datasets, 2018.

[25] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *arXiv*, pages arXiv–1912, 2019.

[26] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[27] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018.

[28] Qianwen Chao, Huikun Bi, Weizi Li, Tianlu Mao, Zhaoqi Wang, Ming C. Lin, and Zhigang Deng. A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. *Computer Graphics Forum*, 39(1):287–308, 2019.

[29] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[30] Udacity's Self-Driving Car Simulator, https://github.com/udacity/self-driving-car-sim, 2017.

[31] NVIDIA DRIVE™ Constellation AV Simulator, https://www.nvidia.com/en-us/self-driving-cars/drive-constellation/, 2019.

[32] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[33] Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, Zeyu Chen, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. *arXiv preprint arXiv:1901.09229*, 2019.

[34] Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[35] Zhi Kou, Kaichao You, Mingsheng Long, and Jianmin Wang. Stochastic normalization. *Advances in Neural Information Processing Systems*, 33:16304–16314, 2020.

[36] Kaichao You, Zhi Kou, Mingsheng Long, and Jianmin Wang. Co-tuning for transfer learning. *Advances in Neural Information Processing Systems*, 33:17236–17246, 2020.

[37] Jincheng Zhong, Ximei Wang, Zhi Kou, Jianmin Wang, and Mingsheng Long. Bi-tuning of pre-trained representations. *arXiv preprint arXiv:2011.06182*, 2020.

[38] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)*, 2020.

[39] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[40] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[41] Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017.

[42] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[43] Sahand Sharifzadeh, Ioannis Chiotellis, Rudolph Triebel, and Daniel Cremers. Learning to drive using inverse reinforcement learning and deep q-networks. *arXiv preprint arXiv:1612.03653*, 2016.

[44] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114:1–18, 2019.

[45] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

[46] Yang Wang. Survey on deep multi-modal data analytics: Collaboration, rivalry, and fusion. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(1s):1–25, 2021.

[47] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[48] Weizi Li, David Wolinski, and Ming C. Lin. ADAPS: Autonomous driving via principled simulations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7625–7631, 2019.

[49] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[50] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[51] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016.

[52] Yiren Zhou, Sibo Song, and Ngai-Man Cheung. On classification of distorted images with deep convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1213–1217. IEEE, 2017.

[53] Samuel Dodge and Lina Karam. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119*, 2017.

[54] Samuel Dodge and Lina Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–7. IEEE, 2017.

[55] Camilo Pestana, Naveed Akhtar, Wei Liu, David Glance, and Ajmal Mian. Adversarial Perturbations Prevail in the Y-Channel of the YCbCr Color Space. *arXiv e-prints*, page arXiv:2003.00883, February 2020.

[56] Zhe Wu, Zuxuan Wu, Bharat Singh, and Larry Davis. Recognizing instagram filtered images with feature de-stylization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:12418–12425, 04 2020.

[57] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. MaxUp: A Simple Way to Improve Generalization of Neural Network Training. *arXiv e-prints*, page arXiv:2002.09024, February 2020.

[58] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[59] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. 2020.

[60] Vasili Ramanishka, Yi-Ting Chen, Teruhisa Misu, and Kate Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7699–7707, 2018.

[61] Isabelle Bégin and Frank Ferrie. Blind super-resolution using a learning-based approach. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 85–89, 2004.

[62] Zhengyou Zhang. A flexible new technique for camera calibration. *Technical Report MSR-TR-98-71, Microsoft Research*, 1998.

[63] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[64] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[65] Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590, 2002.

[66] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[67] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142. IEEE, 2018.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Manli Shu, Zuxuan Wu, Micah Goldblum, and Tom Goldstein. Preparing for the worst: Making networks less brittle with adversarial batch normalization. *arXiv preprint arXiv:2009.08965*, 2020.

[70] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[72] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[73] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[74] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[75] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[76] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[77] Frederick Tung, Jianhui Chen, Lili Meng, and James J Little. The raincouver scene parsing benchmark for self-driving in adverse weather and at night. *IEEE Robotics and Automation Letters*, 2(4):2188–2193, 2017.

[78] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[79] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous, and James McBride. Ford multi-av seasonal dataset. *arXiv preprint arXiv:2003.07969*, 2020.

[80] Matthew Pitropov, Danson Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *arXiv preprint arXiv:2001.10117*, 2020.

[81] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. Deep learning-enabled medical computer vision. *NPJ digital medicine*, 4(1):1–9, 2021.

[82] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational Visual Media*, pages 1–38, 2022.

[83] Junyi Chai, Hao Zeng, Anming Li, and Eric WT Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021.

[84] Xingyu Jiang, Jiayi Ma, Guobao Xiao, Zhenfeng Shao, and Xiaojie Guo. A review of multimodal image matching: Methods and applications. *Information Fusion*, 73:22–71, 2021.

[85] Gargi Joshi, Rahee Walambe, and Ketan Kotecha. A review on explainability in multimodal deep neural nets. *IEEE Access*, 2021.

[86] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[87] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836, 2016.

[88] Woojeong Jin, Maziar Sanjabi, Shaoliang Nie, Liang Tan, Xiang Ren, and Hamed Firooz. Modality-specific distillation. *arXiv preprint arXiv:2101.01881*, 2021.

[89] Nuno Cruz Garcia, Sarah Adel Bargal, Vitaly Ablavsky, Pietro Morerio, Vittorio Murino, and Stan Sclaroff. Distillation multiple choice learning for multimodal action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2755–2764, 2021.

[90] Zhifan Gao, Jonathan Chung, Mohamed Abdelrazek, Stephanie Leung, William Kongto Hau, Zhanchao Xian, Heye Zhang, and Shuo Li. Privileged modality distillation for vessel border detection in intracoronary imaging. *IEEE transactions on medical imaging*, 39(5):1524–1534, 2019.

[91] Kang Li, Lequan Yu, Shujun Wang, and Pheng-Ann Heng. Towards cross-modality medical image segmentation with online mutual knowledge distillation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 775–783, 2020.

[92] Yefeng Zheng. Cross-modality medical image detection and segmentation by transfer learning of shapel priors. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 424–427. IEEE, 2015.

[93] Vanya V Valindria, Nick Pawlowski, Martin Rajchl, Ioannis Lavdas, Eric O Aboagye, Andrea G Rockall, Daniel Rueckert, and Ben Glocker. Multi-modal learning from unpaired images: Application to multi-organ segmentation in ct and mri. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 547–556. IEEE, 2018.

[94] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[95] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Shu-Tao Xia. Adaptive regularization of labels. *arXiv preprint arXiv:1908.05474*, 2019.

[96] Tiancheng Wen, Shenqi Lai, and Xueming Qian. Preparing lessons: Improve knowledge distillation with better supervision. *arXiv preprint arXiv:1911.07471*, 2019.

[97] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *International Conference on Learning Representations (ICLR)*, 2015.

[98] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. *Advances in Neural Information Processing Systems (NIPS)*, pages 2765–2774, 2018.

[99] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1345–1354, 2019.

[100] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4320–4328, 2018.

[101] Xijun Wang, Dongyang Liu, Meina Kan, Chunrui Han, Zhongqin Wu, and Shiguang Shan. Triplet knowledge distillation. *arXiv preprint arXiv:2305.15975*, 2023.

[102] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[103] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[104] Ming-Wei Chang, Vivek Srikumar, Dan Goldwasser, and Dan Roth. Structured output learning with indirect supervision. In *ICML*, pages 199–206, 2010.

[105] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020.

[106] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1365–1374, 2019.

[107] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5007–5016, 2019.

[108] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[109] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Probabilistic knowledge transfer for lightweight deep representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2030–2039, 2020.

[110] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[111] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3779–3787, 2019.

[112] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. *arXiv preprint arXiv:1802.04977*, 2018.

[113] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.

[114] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. *Advances in neural information processing systems*, 32:35–45, 2019.

[115] Xavier Soria Poma, Edgar Riba, and Angel Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1923–1932, 2020.

[116] Zongbo Han, Changqing Zhang, Huazhu Fu, and Joey Tianyi Zhou. Trusted multi-view classification. *arXiv preprint arXiv:2102.02051*, 2021.

[117] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.

[118] Justin Wilson, Nicholas Rewkowski, and Ming C Lin. Audio-visual depth and material estimation for robot navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9239–9246. IEEE, 2022.

[119] Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters*, 7(4):10914–10921, 2022.

[120] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. What is the fast fourier transform? *Proceedings of the IEEE*, 55(10):1664–1674, 1967.

[121] Yu Shen, Laura Zheng, Manli Shu, Weizi Li, Tom Goldstein, and Ming C. Lin. Gradient-free adversarial training against image corruption for learning-based steering. In *Neural Information Processing Systems (NIPS)*, 2021.

[122] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[123] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

[124] UCI. Multiple Features Data Set, https://archive.ics.uci.edu/ml/datasets/multiple+features, 0.

[125] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[126] Yiming Li, Shunli Ren, Pengxiang Wu, Siheng Chen, Chen Feng, and Wenjun Zhang. Learning distilled collaboration graph for multi-agent perception. *Advances in Neural Information Processing Systems*, 34:29541–29552, 2021.

[127] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: Multi-agent perception via communication graph grouping. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 4106–4115, 2020.

[128] Yen-Cheng Liu, Junjiao Tian, Chih-Yao Ma, Nathan Glaser, Chia-Wen Kuo, and Zsolt Kira. Who2com: Collaborative perception via learnable handshake communication. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6876–6883. IEEE, 2020.

[129] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[130] Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 247–263. Springer, 2020.

[131] Zheng Li, Xiang Li, Lingfeng Yang, Borui Zhao, Renjie Song, Lei Luo, Jun Li, and Jian Yang. Curriculum temperature for knowledge distillation. *arXiv preprint arXiv:2211.16231*, 2022.

[132] Wenhao Chai and Gaoang Wang. Deep vision multimodal learning: Methodology, benchmark, and trend. *Applied Sciences*, 12(13):6588, 2022.

[133] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250*, 2017.

[134] Ming Hou, Jiajia Tang, Jianhai Zhang, Wanzeng Kong, and Qibin Zhao. Deep multimodal multilinear fusion with high-order polynomial pooling. *Advances in Neural Information Processing Systems*, 32, 2019.

[135] Ran Xu, Caiming Xiong, Wei Chen, and Jason Corso. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[136] Amir Zadeh, Paul Pu Liang, Navonil Mazumder, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Memory fusion network for multi-view sequential learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[137] Nan Xu, Wenji Mao, and Guandan Chen. Multi-interactive memory network for aspect based multimodal sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 371–378, 2019.

[138] Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018.

[139] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018.

[140] Hong Chen, Xin Wang, Chaoyu Guan, Yue Liu, and Wenwu Zhu. Auxiliary learning with joint task and data scheduling. In *International Conference on Machine Learning*, pages 3634–3647. PMLR, 2022.

[141] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[142] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[143] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[144] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning*, pages 1607–1616. PMLR, 2018.

[145] Jue Jiang, Andreas Rimner, Joseph O Deasy, and Harini Veeraraghavan. Unpaired cross-modality educed distillation (cmedl) applied to ct lung tumor segmentation. *arXiv preprint arXiv:2107.07985*, 2021.

[146] Qi Dou, Quande Liu, Pheng Ann Heng, and Ben Glocker. Unpaired multimodal segmentation via knowledge distillation. *IEEE transactions on medical imaging*, 39(7):2415–2425, 2020.

[147] Zhengyuan Yang, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo. End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2289–2294. IEEE, 2018.

[148] Mohamed Abou-Hussein, Stefan H Müller, and Joschka Boedecker. Multi-modal spatio-temporal information in end-to-end networks for automotive steering prediction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8641–8647. IEEE, 2019.

[149] Sebastian Huch, Aybike Ongel, Johannes Betz, and Markus Lienkamp. Multi-task end-to-end self-driving architecture for cav platoons. *Sensors*, 21(4):1039, 2021.

[150] Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[151] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *ICCV*, volume 3, pages 487–493, 2003.

[152] Manli Shu, Yu Shen, Ming C Lin, and Tom Goldstein. Adversarial differentiable data augmentation for autonomous systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14069–14075. IEEE, 2021.

[153] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[154] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[155] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[156] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[157] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[158] Martijn van Breukelen, Robert PW Duin, David MJ Tax, and JE Den Hartog. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):381–386, 1998.

[159] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. How useful is photo-realistic rendering for visual learning? In *European Conference on Computer Vision*, pages 202–217. Springer, 2016.

[160] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[161] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[162] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[163] W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, W. W. Xu, G. P. Wang, D. Manocha, and R. G. Yang. AADS: Augmented autonomous driving simulation using data-driven algorithms. *Science Robotics*, 4(28), 2019.

[164] American Honda Motor Co. 2014 Civic Si specifications and features, https://hondanews.com/en-us/honda-automobiles/releases/release-b228c382366b432890e04499ebb2b995-2014-civic-si-specifications-and-features, 2014.

[165] Jason Brownlee. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

[166] Mohammadreza Iman, Hamid Reza Arabnia, and Khaled Rasheed. A review of deep transfer learning and recent advancements. *Technologies*, 11(2):40, 2023.

[167] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[168] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[169] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 819–828, 2020.

[170] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[171] Yu Shen, Xijun Wang, Peng Gao, and Ming C Lin. Auxiliary modality learning with generalized curriculum distillation. 2023.

[172] Yu Shen, Luyu Yang, Xijun Wang, and Ming C Lin. Small-shot multi-modal distillation for vision-based autonomous steering. 2023.

[173] Bo Li, Yifei Shen, Yezhen Wang, Wenzhen Zhu, Dongsheng Li, Kurt Keutzer, and Han Zhao. Invariant information bottleneck for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7399–7407, 2022.

[174] Kaiwen Yang, Yanchao Sun, Jiahao Su, Fengxiang He, Xinmei Tian, Furong Huang, Tianyi Zhou, and Dacheng Tao. Adversarial auto-augment with label preservation: A representation learning principle guided approach. *Advances in Neural Information Processing Systems*, 35:22035–22048, 2022.

[175] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[176] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

[177] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[178] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *International conference on machine learning*, pages 1081–1090. PMLR, 2019.

[179] Shimon Ullman. Against direct perception. *Behavioral and Brain Sciences*, 3(3):373–381, 1980.

[180] Andreea Bobu, Andrea Bajcsy, Jaime F Fisac, Sampada Deglurkar, and Anca D Dragan. Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics*, 36(3):835–854, 2020.

[181] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[182] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

[183] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[184] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017.

[185] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[186] Yu Shen, Laura Zheng, Manli Shu, Weizi Li, Tom Goldstein, and Ming C. Lin. Gradient-free adversarial training against image corruption for learning-based steering. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[187] Moritz Werling, Sören Kammel, Julius Ziegler, and Lutz Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, 2012.

[188] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[189] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR, 2018.

[190] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[191] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

[192] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[193] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *The International Conference on Learning Representations (ICLR)*, 2018.

[194] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[195] Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D Dragan. Inducing structure in reward learning by learning features. *arXiv preprint arXiv:2201.07082*, 2022.

[196] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[197] Terence D Sanger. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, 10(3):323–333, 1994.

[198] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, pages 2536–2542, 2017.

[199] Yunlong Song, HaoChih Lin, Elia Kaufmann, Peter Dürr, and Davide Scaramuzza. Autonomous overtaking in gran turismo sport using curriculum reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9403–9409. IEEE, 2021.

[200] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020.

[201] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[202] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[203] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

[204] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[205] David Wilkie, Jason Sewall, Weizi Li, and Ming C. Lin. Virtualized traffic at metropolitan scales. *Frontiers in Robotics and AI*, 2:11, 2015.

[206] Weizi Li, Dong Nie, David Wilkie, and Ming C. Lin. Citywide estimation of traffic dynamics via sparse GPS traces. *IEEE Intelligent Transportation Systems Magazine*, 9(3):100–113, 2017.

[207] Weizi Li, David Wolinski, and Ming C. Lin. City-scale traffic animation using statistical learning and metamodel-based optimization. *ACM Trans. Graph.*, 36(6):200:1–200:12, November 2017.

[208] Lei Lin, Weizi Li, Huikun Bi, and Lingqiao Qin. Vehicle trajectory prediction using LSTMs with spatial-temporal attention mechanisms. *IEEE Intelligent Transportation Systems Magazine*, 2021.