Task-Driven Domain-Agnostic Learning for Autonomous Steering

Anonymous Author(s)
Affiliation
Address
email

Abstract

Autonomous vehicles (AVs) offer the potential for safer and more efficient transportation systems. However, environments for autonomous driving can drastically vary from place to place, which leads to challenges in applying a learning model to a new scene. To address this problem, transfer learning is designed to leverage knowledge from a learned domain to a new domain with limited knowledge. In this work, we focus on end-to-end autonomous driving as the target task, consisting of both perception and control. We first analyze how training data, network architecture, and training paradigm influence the end-to-end steering task, then we propose a novel domain-agnostic learning method for autonomous steering, based on our analysis from those three perspectives. Experiments show that our method outperforms other SOTA methods.

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 [25], 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 [13], NVIDIA [6], Waymo Open Dataset [35], CityScapes [10], and BDD100K [40]. In addition to real-world images, simulators and virtual images are also heavily used in training AVs [5]. Example simulation platforms include CARLA [11], the Udacity Self-Driving Car Simulator [1], and NVIDIA Drive Constellation [2]. 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 [36] such as CycleGAN [42] and MUNIT [19] challenges the domain gap and has raised new conjectures on whether we can use the realistic-looking images converted from virtual images for learning [31]. 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

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

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. 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.

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.

2 Related Work

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 [29] is able to learn in the target domain while keeping the memory of the source domain without source domain data. DELTA [28] proposes a novel regularized transfer learning framework, preserving the outer layer outputs of the target network. BSS [7] presents a novel regularization approach to penalizing smaller singular values so that untransferable spectral components are suppressed. StochNorm [24] proposes a two-branch design with one branch normalized by mini-batch statistics and the other branch normalized by moving statistics. Co-Tuning [39] 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 [41] 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.

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. [30] 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 [22]. Among many variants of GAN, CycleGAN [42] has been successfully applied to style-transfer unpaired images in training data. CyCADA [16], 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. [34] 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. [26] 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.

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 [6] as our real dataset, which contains approximately 63,000 images at the resolution 455×256. The data is recorded on urban/suburban roads in California. We use the data from the Udacity Self-Driving Car Simulation [1] as our virtual dataset, which includes about 10,615 images at the resolution 320×160 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 [9], 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. 2(a) and Fig. 2(b), respectively.

Evaluation Metric. We use mean accuracy (MA) to evaluate our regression 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. [3] 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 [3] and virtual world [27].

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.

- $\mathrm{train}(A + B)$: simply combine datasets A and B and use the combined dataset for training;
- $\mathrm{train}(A) \rightarrow \mathrm{train}(B)$: use A to pretrain a model, then use B to retrain the model; and
- $\mathrm{train}(A) \rightarrow \mathrm{ptrain}(B)$: 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 [4]. Since we are using the model by Bojarski et al. [3], 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.

4 Analysis

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

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 [42] and MUNIT [19]. We show sample images of the two types of style-transferred images in Fig. 2(c) and (d), respectively.

Table 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.

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

- R: real dataset (the Nvidia dataset [6], target domain);
- V: virtual dataset (the Udacity dataset [1]);
- T_C, T_M: style-transferred datasets from virtual to real using CycleGAN [42] and MUNIT [19], 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 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, $\mathrm{MA}_R(\mathrm{train}(R))$ - $\mathrm{MA}_R(\mathrm{train}(T_C))$ = 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. 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 1(a): $\mathrm{MA}_R(\mathrm{train}(T_C)) < \mathrm{MA}_R(\mathrm{train}(V)) < \mathrm{MA}_R(\mathrm{train}(R))$, and $\mathrm{MA}_R(\mathrm{train}(T_M)) < \mathrm{MA}_R(\mathrm{train}(V)) < \mathrm{MA}_R(\mathrm{train}(R))$. 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. 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 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).

Table 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.

	Train					
Test	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%*

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

In addition, we try to evaluate the domain gap with Fréchet Inception Distance (FID) [15], 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 7 (in Appendix 7.2), the relative order is different from the actual test results in Table 2.

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 [20] 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 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 [14]), virtual (V), or style transferred (T_C, T_M) datasets as source domain.

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 [21]. 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** [17]. 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 4, we compare normal BN [21], AdvProp BN [38], and LoRA [18] under the best training paradigms in previous experiments. **LoRA achieves the best performance in the list.**

Table 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**.

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

Table 4: Mean Accuracy comparison with different network architectures. **LoRA outperform others**.

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

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 6 (e) (f) verifies this hypothesis.

5 Our Method

Inspired by the analysis in Sec. 4, we design our framework from three perspectives, i.e., training data, network architecture, and training paradigm.

Table 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)).

	Model (M)	$\mathrm{MA}_R(M)$
(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%

5.1 Framework

Overview. Our framework overview is shown in Fig. 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. 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 [38] 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 [24].

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 [32, 33], we can add "hint" images in

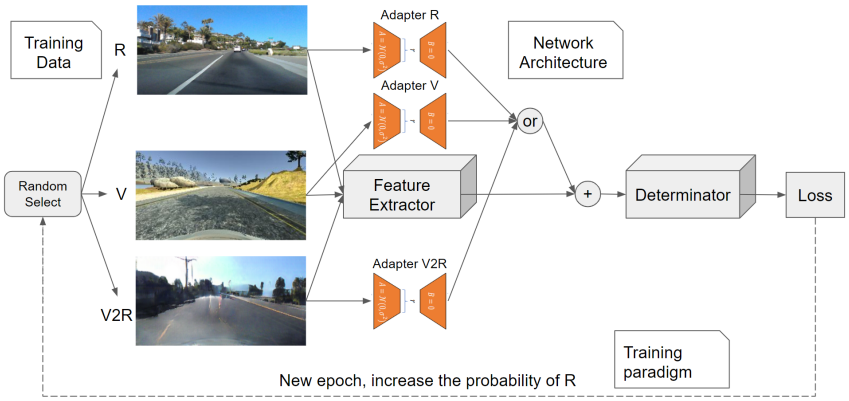
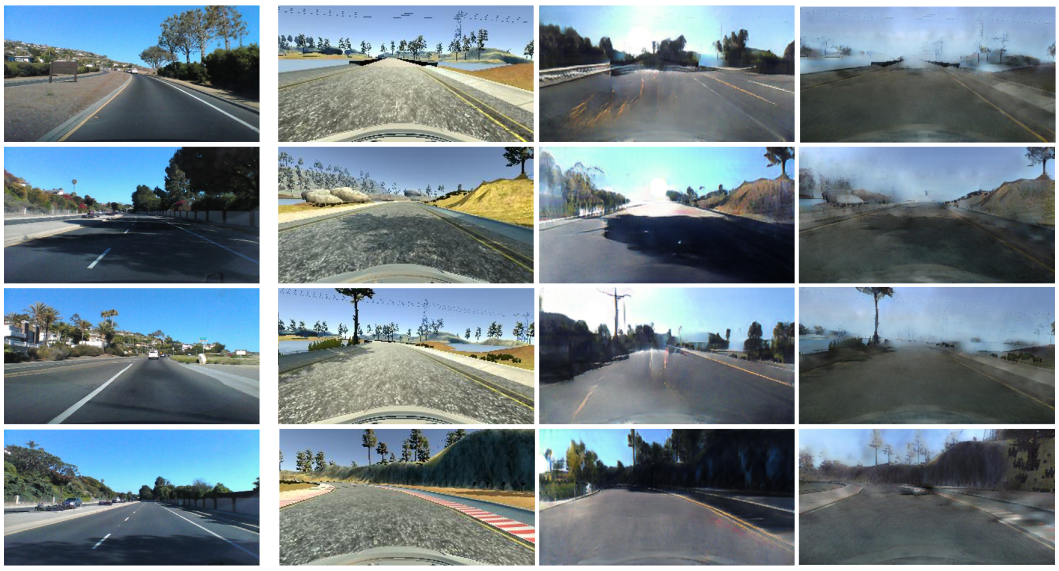


Figure 1: **Our framework**: In each epoch, the input data is randomly selected from real, virtual, and style-transferred datasets, which will be fed into a shared feature extractor and a domain-dependent adapter. The combined output feature will then be used to determine the final steering angle. The 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.

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 [42], 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. 2.



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

Figure 2: Sample images of various datasets. (a) the Nvidia dataset [6] (real dataset, denoted by R). (b) the Udacity dataset [1] (virtual dataset, denoted by V). (c) style-transferred images from virtual to real using CycleGAN [42] (denoted by T_C). (d) style-transferred images from virtual to real using MUNIT [19] (denoted by T_M). We plan to release all datasets for comparative experiments.

Design in training paradigm. In Sec. 4.2, we show that **finetuning is 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

Table 6: Mean Accuracy comparison with domain adaptation and task adaptation methods, and ablation study. Our method outperforms others under all angle thresholds. LoRA is the adapter, STB stands for style transferred branch, DP stands for dynamic probability for each domain.

		$\mathrm{MA}_R(M)$ (%) on different angle threshold τ (degree)						
	Method	$\tau = 1.5$	$\tau = 3.0$	$\tau = 7.5$	$\tau = 15$	$\tau = 30$	$\tau = 75$	**mAcc**
	Baseline	59.5%	82.1%	93.9%	96.3%	98.6%	99.8%	88.36%
(a) Domain Adaptation	DANN [12]	38.1%	59.8%	75.3%	88.7%	93.1%	95.9%	75.15%
	ADDA [37]	40.6%	61.7%	78.4%	89.8%	94.4%	97.5%	77.06%
	BSP [8]	43.3%	66.9%	83.6%	92.4%	95.8%	98.9%	80.15%
(b) Task Adaptation	DELTA [28]	60.1%	83.9%	94.3%	96.7%	98.6%	99.9%	88.91%
	BSS [7]	63.9%	85.7%	94.9%	97%	98.8%	99.9%	90.03%
	StochNorm [24]	57.1%	80.6%	91.9%	95.2%	97.5%	99.2%	86.91%
(c) Ablation	Ours w/o LoRA	60.2%	81.9%	92.1%	95.8%	98.6%	99.8%	88.06%
	Ours w/o STB	64.6%	85.9%	94.5%	96.9%	98.6%	99.9%	90.06%
	Ours w/o DP	63.7%	83.9%	94.1%	96.9%	98.7%	99.9%	89.53%
	Ours	**65.5%**	**86.9%**	**95.3%**	**97.1%**	**98.9%**	99.9%	**90.60%**

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 is a straightforward L2 loss between the output of the network and the ground-truth steering angle.

5.2 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 [23] 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. 3.

Comparison with other task adaptation methods. We compare our method with other SOTA task adaptation methods, i.e., DELTA [28], BSS [7], StochNorm [24]. 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. We compare our method with other classic domain adaptation methods, i.e., DANN [12], ADDA [37], BSP [8]. 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) takes effect, we do an ablation study on each of them. Results show that removing any component will lead to a performance drop, which means each of them does contribute to the final performance.

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.

References

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

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

[3] 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.

[4] Jason Brownlee. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Mohammadreza Iman, Hamid Reza Arabnia, and Khaled Rasheed. A review of deep transfer learning and recent advancements. *Technologies*, 11(2):40, 2023.

[21] 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.

[22] 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.

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

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

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

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

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

[32] Yu Shen, Xijun Wang, Peng Gao, and Ming C Lin. Auxiliary modality learning with generalized curriculum distillation. 2023.

[33] Yu Shen, Luyu Yang, Xijun Wang, and Ming C Lin. Small-shot multi-modal distillation for vision-based autonomous steering. 2023.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

[42] 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.

7 Appendix

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.

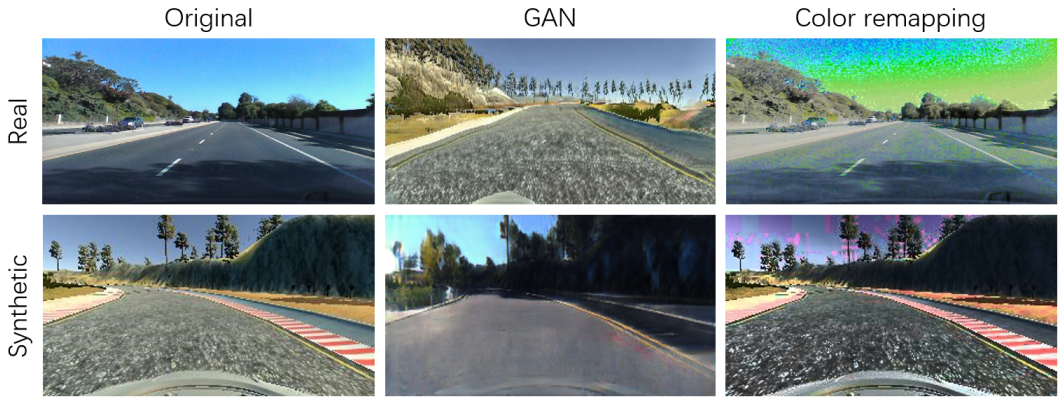


Figure 3: We show sample images from the datasets R (Real photos in Nvidia dataset), (Real photos in virtual style generated by GAN), (Real photos in virtual style generated by color remapping) in the first row from left to right, and V (Virtual images in Udacity dataset), T_C (Virtual images in real style generated by GAN), (Virtual images in real style generated by color remapping) in the second row from left to right.

7.2 Fréchet Inception Distance

We try to evaluate the domain gap with Fréchet Inception Distance (FID) [15], 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 7 (in Appendix 7.2), the relative order is different from the actual test results in Table 2.

Table 7: Fréchet Inception Distance between different datasets.

	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