Real-to-Virtual Domain Unification for End-to-End Autonomous Driving

Luona Yang

CMU-RI-TR-18-07

May 15, 2018

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Eric P. Xing, Advisor
John M. Dolan
Devin Schwab

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Robotics.

Abstract

In the spectrum of vision-based autonomous driving, vanilla end-to-end models are not interpretable and are suboptimal in performance, while mediated perception models require additional intermediate representations such as segmentation masks or detection bounding boxes, whose annotation can be prohibitively expensive as we move to a larger scale. More critically, all prior works fail to deal with the notorious domain shift if we were to merge data collected from different sources, which greatly hinders the model generalization ability. In this work, we address the above limitations by taking advantage of virtual data collected from driving simulators, and present DU-drive, an unsupervised real-to-virtual domain unification framework for end-to-end autonomous driving. It first transforms real driving data to its less complex counterpart in the virtual domain, and then predicts vehicle control commands from the generated virtual image. Our framework has three unique advantages: 1) it maps driving data collected from a variety of source distributions into a unified domain, effectively eliminating domain shift; 2) the learned virtual representation is simpler than the input real image and closer in form to the "minimum sufficient statistic" for the prediction task, which relieves the burden on the compression phase while optimizing the information bottleneck tradeoff and leads to superior prediction performance; 3) it takes advantage of annotated virtual data which are unlimited and free to obtain. Extensive experiments on two public driving datasets and two driving simulators clearly demonstrate the performance superiority and interpretive capability of DU-drive.

Acknowledgments

Firstly, I want to thank my advisor, Professor Eric Xing, for offering me the opportunity to intern at his lab three years ago, at which time I still knew very little about machine learning or robotics, and for inspiring me to pursue the subjects I love no matter how humble the starting point is. I also want to thank Professor John Dolan and Mr. Devin Schwab for serving on my thesis committee and providing many insightful feedbacks. I'm humbled by their dedication to perfection and aspire to live up to the standards they have set in my future work.

I must thank Xiaodan Liang, for the many interesting discussions we have had, and the incredible support she has offered me in building up my thesis. Our soon-to-be professor will make a great teacher and a great researcher.

I also want to extend my gratitude to the many friends I have made here at CMU. Their friendship and invaluable support in my difficult times have brought me great joy during my time here.

Of course, none of this would happen without the support and love from my family. I could not ask for better parents, who have given me unconditional love, trust, and encouragement throughout my life.

Finally, special thanks to my boyfriend, Hao Zhang, not only for his constant love and support, but also for his lighthearted pranks and for always making me smile. We have had so much fun together. My life would have lost its luster without you.

Contents

List of Figures

List of Tables

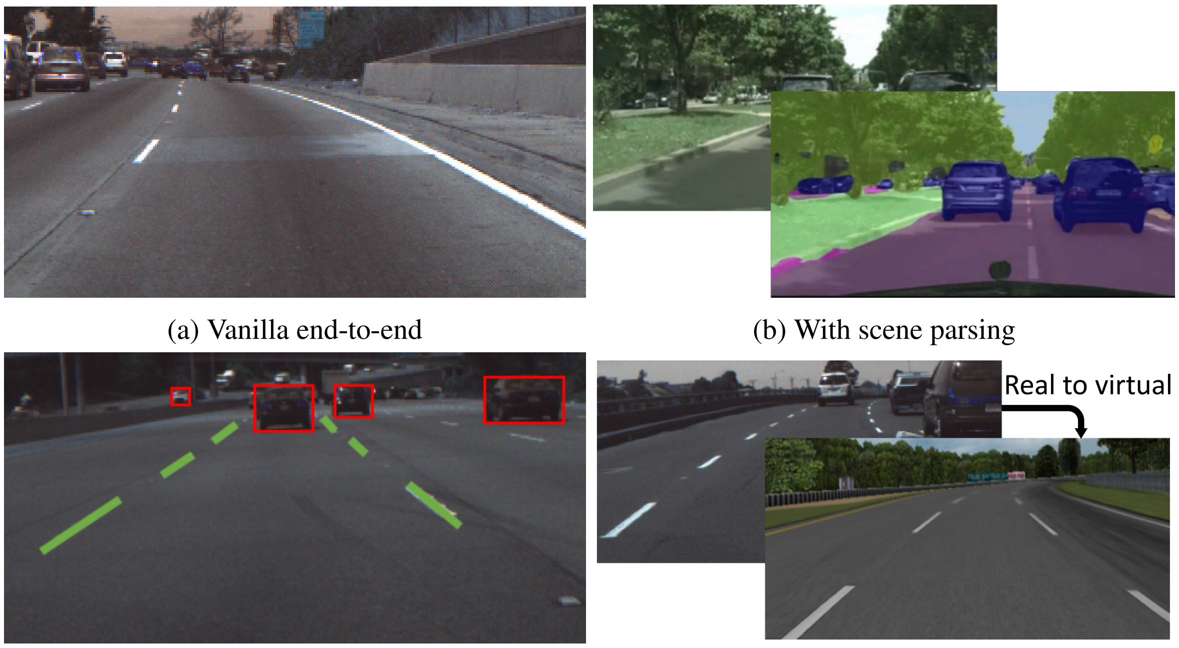
Chapter 1

Introduction

The development of a vision-based autonomous driving system has been a long-standing research problem [2, 3, 4, 5]. End-to-end models, among many methods, have attracted much research interest [6, 7, 8] as they optimize all intermediate procedures simultaneously and eliminate the tedious process of feature engineering. [6] trains a convolutional neural network (CNN) to map raw image pixels from a frontal camera to steering commands, which successfully maneuvered the test car in constrained environments. Many attempts have since been made to improve the performance of vanilla end-to-end models by taking advantage of intermediate representations (Figure 1.1). For example, [7] uses semantic segmentation as a side task to improve model performance, while [9] first trains a detector to detect nearby vehicles before making driving decisions. However, the collection of driving data and the annotation of intermediate representation can be prohibitively expensive as we move to a larger scale.

Moreover, raw images of driving scenes are loaded with nuisance details that are not relevant to the prediction task due to the complexity of the real world. For example, a typical human driver will not change his or her behavior according to the shadow of trees on the road, or the view beyond the road boundaries. Such nuisance information could distract the neural network from what is truly important and negatively impact prediction performance. [10] visualizes the activation of the neural network and shows that the model not only learns driving-critical information such as lane markings, but also unexpected features such as atypical vehicle classes. [8] presents results of the driving model's attention map refined by causal filtering, which seems to include rather random attention blobs.

As pointed out by [11] in the information bottleneck principle, the learning objective for a deep neural network could be formulated as finding the optimal representation that maximally compresses the information in the input while preserving as much information as possible about the output, or in other words, finding an approximate minimal sufficient statistic of the input with respect to the output. Further work [12] shows that the Stochastic Gradient Descent (SGD) optimization of the neural network has two distinct phases, the fitting phase during which the mutual information of the intermediate layers with the output increases and empirical error drops, and the compression phase during which the mutual information of the intermediate layers with the input decreases and the representation becomes closer in form to the minimum sufficient statistic of the output. They also show that most of the training effort is spent on the compression phase, which is the key to good generalization. It is therefore beneficial for the optimization of



(a) Vanilla end-to-end

(b) With scene parsing

(c) With vehicle/lane detection

(d) With DU-drive (ours)

Figure 1.1: Various methods have been proposed for vision-based driving models. While vanilla end-to-end models (a) are not interpretable and are suboptimal in performance, scene parsing (b) or object detection (c) requires expensively annotated data. Our method (d) unifies real images from different datasets into their simpler counterparts in the virtual domain that contains fewer superfluous details, which boosts the performance of the vehicle command prediction task.

the network to have a representation that contains less irrelevant complexity, as it could relieve the burden of the compression phase by giving a better "initialization" of the optimal representation.

More critically, all existing work focuses on a single source of data and does not explicitly deal with generalization to an unseen dataset. As noted by [13], datasets could have strong built-in biases, and a well-functioning model trained on one dataset will very likely not work so well on another dataset that is collected differently. This phenomenon is known as domain shift, which characterizes the distance in the distribution of inputs and outputs from different domains. While the existing model could be tuned to gradually fit the new domain with the injection of more and more supervised data from the new environment, this could be extremely data-inefficient and prohibitively expensive for tasks with diverse application scenarios like autonomous driving.

We propose to tackle the above challenges by taking advantage of virtual data collected from simulators. Our DU-drive system maps real driving images collected under variant conditions into a unified virtual domain, and then predicts vehicle commands from the generated fake virtual image. Since all real datasets are mapped to the same domain, we could easily extend our model to unseen datasets while taking full advantage of the knowledge learned from existing ones. Moreover, virtual images are "cleaner" as they are less complex and contain less noise, and thus closer to the "minimal sufficient statistic" of the vehicle command prediction task, which is the target representation that the neural network should learn under the information bottleneck framework. Last but not least, our model could make full use of unlimited virtual data and the simulation environment, and a model learned in the virtual environment could be directly applied to a new dataset after unifying it to the virtual domain. Experimental results on two public driving datasets and two driving simulators under supervised and semi-supervised settings, together with analysis of the efficiency of the learned virtual representation compared to raw image input under the information bottleneck framework clearly demonstrate the performance superiority of our method.

Chapter 2

Related Work

2.1 Vision-based Autonomous Driving

Vision-based solutions are believed to be a promising direction for solving autonomous driving due to their low sensor cost and recent developments in computer vision. Since the first successful demonstration in the 1980s [2, 4, 5], various methods have been investigated in the spectrum of vision-based driving models, from end-to-end methods to full pipeline methods [14]. The ALVINN system [3], first introduced in 1989, is the pioneering work in end-to-end learning for autonomous driving. It shows that an end-to-end model can indeed learn to steer under simple road conditions. The network architecture has since evolved from the small fully-connected network of ALVINN into convolutional networks used by the DAVE system [15] and then deep models used by the DAVE-2 system [6]. Intermediate representations such as semantic segmentation masks and attention maps are shown to be helpful to improving the performance [7, 8].

Pipeline methods separate the parsing of the scene and the control of the vehicle. [9] first trains a vehicle detector to determine the location of adjacent cars and outputs vehicle commands according to a simple control logic. [16] shows that convolutional neural networks can be used to do lane and vehicle detection at a real-time frame rate. While such methods are more interpretable and controllable, the annotation of intermediate representations can be very expensive.

Our method takes advantage of an intermediate representation obtained from unsupervised training and therefore improves the performance of vanilla end-to-end driving models without introducing any annotation cost.

2.2 Generative Adversarial Networks

Generative adversarial network (GAN) is a new framework for estimating generative models via an adversarial process first proposed by [17]. Given a set of data X drawn from an unknown distribution p_{data}, GAN introduces a noise variable z drawn from some predefined distribution p_z, and parametrizes the mapping from z to x as $G(z; \theta_g)$, where G is a differentiable function represented as a neural network parametrized by θ_g. Another neural network $D(x; \theta_d)$ is also defined, which takes a data point as input and outputs a single scalar. $D(x)$ represents the probability that

D believes x is from the natural dataset X. D is thus called the discriminative network while G is called the generative network. The two networks are trained hand-in-hand by an adversarial process called the minimax game, where D and G tries to maximize and minimize the following objective $V(D, G)$ respectively:

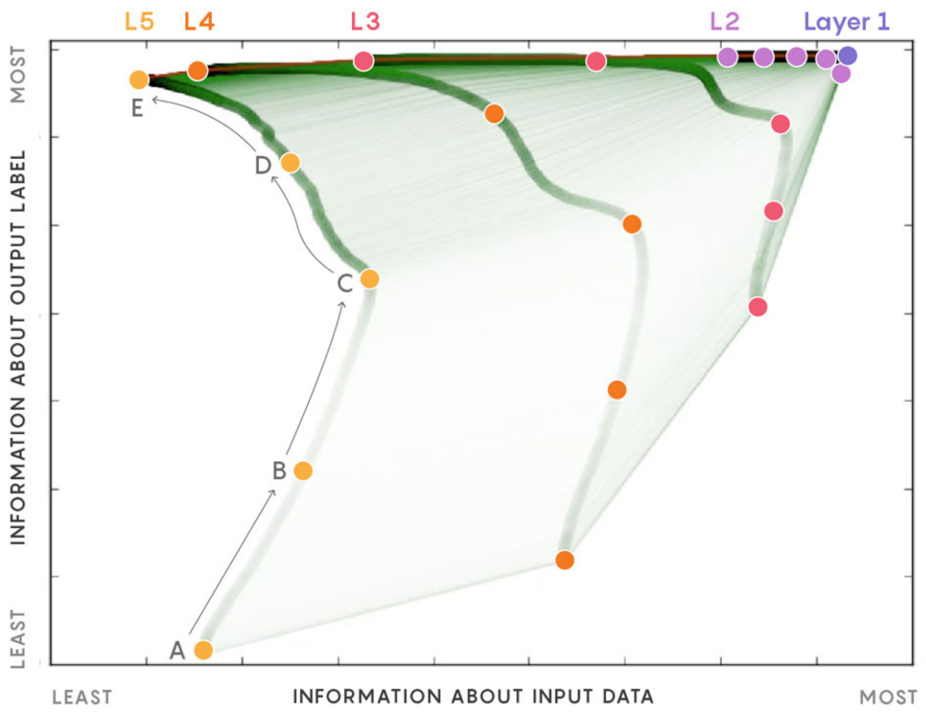
$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

Intuitively, the goal of the discriminator is to distinguish "real" data drawn from the natural data distribution p_{data} and the "fake" data produced by the generative network, while the goal of the generator is to fool the discriminator. During training, the objective is iteratively optimized w.r.t. each network with stochastic gradient-based methods. The training reaches equilibrium when $p_z(G(z)) = p_{data}(x)$, i.e., when the generator generates real-looking data points that the discriminator cannot distinguish from real data points.

However, generative adversarial networks are notoriously difficult to train in their original form. Many variant objectives and training schemes have been proposed to stabilize the training process. [18] showed that minimizing the least square loss for the discriminator is equivalent to minimizing the Pearson divergence between the two distributions. LS-GAN alleviates the gradient vanishing problem and leads to a more stable training process. [19] proposed WGAN, which minimizes the Earth-Mover distance and to a large extent resolves the mode-collapse issue. It also does not require a careful balancing in training the discriminator and the generator. In our implementation, we use LS-GAN to stabilize our training process.

2.3 Information Bottleneck Theorem

Deep learning has celebrated tremendous success in many applications during the past few years, yet little is known about the internal structure or optimization process of deep neural nets. The information bottleneck theorem (IB) [20] suggests the study of the process in the information plane - the plane of mutual information values of the network neurons with the input and output variable. Under the IB framework, [11] discovered that deep learning proceeds in two phases: the "fitting" phase where the mutual information between the network layers and the input/output increases, and the "compression phase" where the mutual information between the network layers and the input decreases. Intuitively, the "fitting" phase could be thought of as the memorization process while the "compression" phase could be thought of as the forgetting process, which is the key to good generalization. The optimization process could be alternatively framed as optimizing the information bottleneck tradeoff, which aims to retain as much information as possible about the output label while discarding irrelevant information in the input. In other words, the goal is to find an estimate of the minimal sufficient statistic of the input with respect to the output. Figure 2.1 illustrates the information path of five layers of a neural network in the information plane during a training process.



A **INITIAL STATE:** Neurons in Layer 1 encode everything about the input data, including all information about its label. Neurons in the highest layers are in a nearly random state bearing little to no relationship to the data or its label.

B **FITTING PHASE:** As deep learning begins, neurons in higher layers gain information about the input and get better at fitting labels to it.

C **PHASE CHANGE:** The layers suddenly shift gears and start to "forget" information about the input.

D **COMPRESSION PHASE:** Higher layers compress their representation of the input data, keeping what is most relevant to the output label. They get better at predicting the label.

E **FINAL STATE:** The last layer achieves an optimal balance of accuracy and compression, retaining only what is needed to predict the label.

Figure 2.1: Illustration of the information paths for a 5-layer neural network during the training process [1].

2.4 Domain Adaptation for Visual Data

Ideally, a model trained for a specific task should be able to generalize to new datasets collected for the same task, yet research has shown that model performance could seriously degrade when input distribution changes due to the inherent bias introduced in the data collection process [13]. This phenomenon is known as domain shift or dataset bias. In the world of autonomous driving, it is even more critical to have a model that can generalize well to unseen scenarios.

Dataset bias originates from many different sources. As pointed out by [13], there is **selection bias** where a dataset often prefers particular kinds of images (e.g. summer scenes with few trees and clear sky for a driving dataset collected in Texas), and **capture bias** caused by a particular setup of cameras (e.g. mounting the camera at a fixed location of a particular vehicle). While it is difficult to measure how badly the dataset bias will degrade the generalization ability of the model, as it is inherently difficult to measure the distance between two high-dimensional distributions, one way to quantify the difference between two datasets is to train a classifier on the two datasets and see how easy it is to distinguish images from different datasets. A pair of datasets that is relatively easy to tell apart is more likely to cause domain shift than a pair of datasets that is relatively difficult to discriminate.

Domain adaptation methods attempt to battle domain shift by bridging the gap between the distribution of source data and target data [21, 22]. Recently, generative adversarial network (GAN) based domain adaptation, also known as adversarial adaptation, has achieved remarkable results in the field of visual domain adaptation. [23] introduces a framework that subsumes several approaches as special cases [24, 25, 26]. It frames adversarial adaptation as training an encoder (generator) that transforms data in the target domain to the source domain at a certain feature level, trying to fool the adversarial discriminator, which in turn tries to distinguish the generated data from those sampled from the source domain. The line of work on style transfer [27, 28, 29] could also be potentially applied to domain adaptation at the pixel level.

One subarea especially to our interest is the adaptation of virtual data to real data. As the collection of real-world data can be excessively expensive in certain cases, virtual data rendered with computer graphics technologies can come to remedy if we could adapt knowledge learned in the virtual domain to the real domain. [30] proposed a GAN-based model that transforms data from the virtual domain to the real domain in the pixel space in an unsupervised manner by utilizing a content-similarity loss to retain annotation. [31] uses adversarial training to improve the realism of synthetic images with the help a self-regularization term, a local adversarial loss and a buffer of training images for the discriminator. [32] randomizes the texture of objects in the robot simulator and trains a visuomotor policy without using any real-world data. [33] trains a driving policy with reinforcement learning in a simulator by transforming virtual images to real images, retaining the scene structure with an adversarial loss on the segmentation mask.

While existing work aims at transforming virtual images into realistic-looking images, we argue that doing the opposite, i.e., transforming real images into the virtual domain, could be more advantageous for learning a driving policy. The transformation from real to virtual is an easier task as it is more manageable to go from complex to simple, and all real datasets could be unified into their simpler counterparts in the virtual domain.

Chapter 3

Unsupervised Domain Unification

3.1 Network Design and Learning Objective

Learning Objective for DU-Drive Given a dataset of driving images labeled with vehicle commands in the real domain and a similar dataset in the virtual domain, our goal is to transform a real image into the virtual domain and then run a prediction algorithm on the transformed fake virtual image. The overall architecture is shown in Figure 3.1. Our model is closely related to conditional GAN [34], where the generator and discriminator both take a conditional factor as input, yet it is different in two subtle respects. One is that in our model, the discriminator does not depend on the conditional factor. The other is that our generator does not take any noise vector as input. Unlike the mapping from a plain virtual image to a rich real image, where there could be multiple feasible solutions, the mapping from a real image to its less complex virtual counterpart should be unique. Therefore, we could remove the noise term in conventional GANs and use a deterministic generative network as our generator.

More formally, let $\mathbf{X}^r = \{\mathbf{x}_i^r, \mathbf{y}_i^r\}_{i=1}^{N_r}$ be a labeled dataset with N^r samples in the real domain, and let $\mathbf{X}^v = \{\mathbf{x}_i^v, \mathbf{y}_i^v\}_{i=1}^{N^v}$ be a labeled dataset with N^v samples in the virtual domain, where $\mathbf{x}$ is the frontal image of a driving scene and $\mathbf{y}$ is the corresponding vehicle command. Our DU-drive model consists of a deterministic conditional generator $G(\mathbf{x}^r; \theta_G) \to \mathbf{x}^f$, parametrized by θ_G, that maps an image $\mathbf{x}^r \in \mathbf{X}^r$ in the real domain to a fake virtual image $\mathbf{x}^f$, a virtual discriminator $D(\mathbf{x}^{\text{v}}; \theta_D)$ that discriminates whether a image is sampled from true virtual images or from fake virtual images, and a predictor $P(\mathbf{x}^v; \theta_P) \to y^v$, that maps a virtual image to a vehicle control command.

The learning objective of DU-drive is:

$$\min_{\theta_G, \theta_P} \max_{\theta_D} \mathcal{L}_d(D, G) + \lambda L_t(P, G), \tag{3.1}$$

where $\mathcal{L}_d(D, G)$ is the domain loss, which the generator tries to minimize and the discriminator tries to maximize in the minimax game of GAN. $\mathcal{L}_d(D, G)$ is defined as:

$$\mathcal{L}_d(D, G) = \mathbb{E}_{\mathbf{x}^v}[\log D(\mathbf{x}^v; \theta_D)] + \tag{3.2}$$

$$\mathbb{E}_{\mathbf{x}^r}[\log(1 - D(G(\mathbf{x}^r; \theta_G); \theta_D))], \tag{3.3}$$

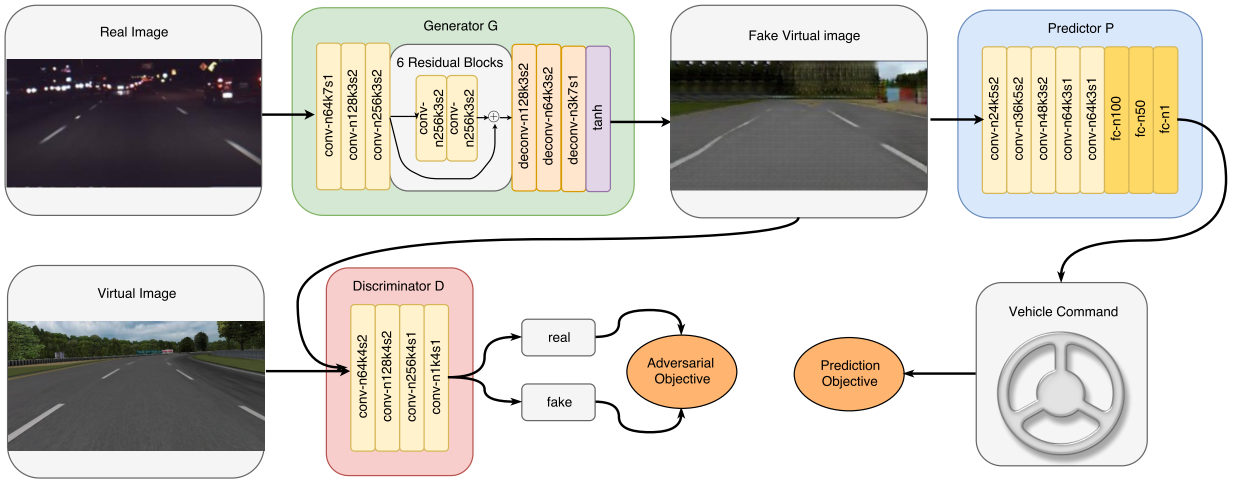


Figure 3.1: Model architecture for DU-Drive. The generator network G transforms input real image to fake virtual image, from which the vehicle command is predicted by the predictor network P. The discriminator network D tries to distinguish the fake virtual images from true virtual images. Both the adversarial objective and the prediction objective drive the generator G to generate the virtual representation that yields the best prediction performance. For simplicity, instance normalization and activation layers after each convolutional/fully connected layer are omitted. (Abbr: n: number of filters, k: kernel size, s: stride size)

$\mathcal{L}_t(P, G)$ is the task-specific objective for predictor and generator, which in this work is the mean square loss between the predicted control command and the ground truth control command, defined as:

$$\mathcal{L}_t(P, G) = \mathbb{E}_{\mathbf{x}^r}[\|P(G(\mathbf{x}^r; \theta_G), \theta_P) - \mathbf{y}^r\|_2^2] \tag{3.4}$$

λ is a hyperparameter that controls the weight of task-specific loss and the domain loss.

Network Design For the GAN part of the model, we mostly adopt the network architecture as defined in [29], which has achieved impressive results in style transfer tasks. The generator network consists of two convolutional layers with 3x3 kernel and stride size 2, followed by 6 residual blocks. Two deconvolutional layers with stride 1/2 then transform the feature to the same size as the input image. We use instance normalization for all the layers. For the discriminator network, we use a fully convolutional network with convolutional layers of filter size 64, 128, 256 and 1. Each convolutional layer is followed by instance normalization and Leaky ReLU nonlinearity. We do not use PatchGAN as employed in [28] because driving command prediction needs global structure information.

For the predictor network, we adopt the network architecture used in the DAVE-2 system[6], also known as *PilotNet*, as it has achieved decent results in end-to-end driving [6, 10, 29]. The network contains 5 convolutional layers and 4 fully connected layers. The first three convolutional layers have kernel size 5x5 and stride size 3, while the last two layers have kernel size 3x3 and stride size 1. No padding is used. The last convolutional layer is flattened and immediately followed by four fully connected layers with output size 100, 50, 10 and 1. All layers use ReLU activation.

3.2 Learning

Our goal is to learn a conditional generator that maps a real image into the virtual domain. However, a naive implementation of conditional GAN is insufficient for two reasons. First, the adversarial loss only provides supervision at the level of image distribution and does not guarantee the retention of the label after transformation. Second, conventional GANs are vulnerable to mode collapse, a common pitfall during the optimization of the GAN objective where the distribution of transformed images degenerates. Previous work on adapting virtual images to real images alleviates those problems by introducing a task-specific loss to add additional constraints to the image generated. For example, [30] uses a content similarity loss to enforce that the foreground of the generated image matches with that of the input image. [31] employs a self-regularization term that minimizes the image difference between the synthetic and refined images.

Unfortunately, we cannot take advantage of similar techniques as the "foreground", or the information critical to retaining the label is not obvious for autonomous driving. Instead, we introduce a joint training scheme, where the conditional generator and the predictor are trained simultaneously, so that the supervision from the prediction task gradually drives the generator to convert the input images from the real domain to its corresponding representation in the virtual domain, which retains the necessary semantics and yields the best prediction performance. More formally, our objective in Eq. 3.1 can be decomposed into three parts with respect to the three networks G, P and D:

$$\min_{\theta_G} \mathcal{L}_d(D, G) + \lambda L_t(P, G), \tag{3.5}$$

$$\min_{\theta_P} L_t(P, G), \tag{3.6}$$

$$\max_{\theta_D} \mathcal{L}_d(D, G) \tag{3.7}$$

We omit the weight term λ in Equation 3.6, as it is easy to see that only θ_G is influenced by both the domain loss and the prediction loss, and we can train θ_D, θ_G and θ_P with respect to the three objectives above independently. We denote α_P as the learning rate for updating θ_P, and α_{GAN} as the learning rate for updating θ_D and θ_G.

During training, we update θ_D, θ_G and θ_P sequentially by alternately optimizing the above three objectives, so that the generation quality and prediction performance improve hand in hand.

3.3 Domain Unification

Consider the case when we have multiple real datasets $\{\mathbf{x}^{r_1}, \mathbf{y}^{r_1}\}$,...,$\{\mathbf{x}^{r_n}, \mathbf{y}^{r_n}\}$. Due to different data distribution depicted by road appearance, lighting conditions or driving scenes, each dataset belongs to a unique domain which we denote as D_{r_1},...,D_{r_n} respectively. Prior works on end-to-end driving tend to deal with only one domain rather than a more general reasoning system. DU-drive, however, unifies data from different real domains into a single virtual domain and eliminates the notorious domain shift problem.

For each real domain D_{r_i}, we use our DU-drive model to train a generator that transforms images $\mathbf{x}^{r_i}$ into their counterparts $\mathbf{x}^{f_i}$ in a unified virtual domain D_v (Figure 3.2). A global predictor P_v could then be trained to do vehicle command prediction from the transformed virtual images. We fix the generator for each real domain and train the global predictor with labeled data from multiple real domains simultaneously. As in our training setup for a single domain, we also use *PilotNet* pretrained on virtual data as our initialization for the global predictor.

3.4 Connection with Information Bottleneck Principle

Given a raw image input, what could be a good intermediate representation that could help boost the performance of the prediction task? We try to answer this question under the information bottleneck framework.

Formally, let X be the raw image input and Y be the vehicle control command that is to be predicted. The information bottleneck objective of learning for a neural network is to find the optimal representation of X w.r.t. Y, which is the minimal sufficient statistic $T(x)$, the simplest sufficient statistic that captures all information about Y in X. However, closed form representation for the minimum sufficient statistic does not exist in general, and according to [12] this objective could be written as a tradeoff between compression of X and prediction of Y formulated in the following form:

$$\mathcal{L}[p(t|x)] = I(X; T) - \beta I(T; Y) \tag{3.8}$$

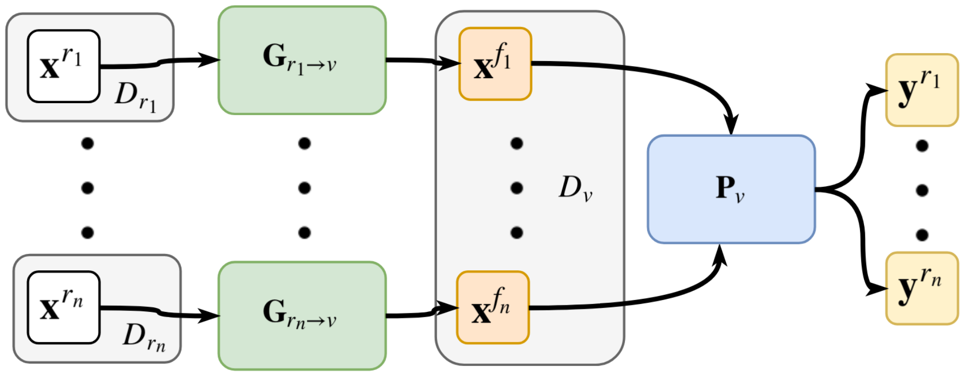


Figure 3.2: Domain unification by DU-drive. For each real domain, a generator is trained independently to transform real images to fake virtual images in a unified virtual domain. A single virtual image to the vehicle command predictor is trained to do prediction across multiple real domains.

where $I(X;T)$ denotes the mutual information between the learned representation and input, and $I(T;Y)$ denotes the mutual information between the learned representation and output. This objective is optimized successively for each layer. At the beginning of training, the objective at the input layer where $T = X$ can be written as

$$
\begin{aligned}
L_{\{T=X\}} =& I(X;X) - \beta I(X;Y) && (3.9)\\
=& H(X) - \beta(H(Y) - H(Y|X)) && (3.10)\\
=& H(X) - \beta H(Y) && (3.11)
\end{aligned}
$$

where Eq. 3.11 follows from the fact that X is a sufficient statistic for Y. Now, consider the case when we have an intermediate representation $G(X)$ of X. We assume that $G(X)$ is also a sufficient statistic of Y, which is reasonable for any meaningful intermediate representation. Then the objective when $T = G(X)$ is

$$
\begin{aligned}
L_{\{T=G(X)\}} =& I(X;G(X)) - \beta I(G(X);Y) && (3.12)\\
=& (H(G(X)) - H(G(X)|X)) - \beta(H(Y) - H(Y|X)) && (3.13)\\
=& H(G(X)) - \beta H(Y) && (3.14)
\end{aligned}
$$

Subtracting Eq. 3.12 from Eq. 3.9 yields:

$$
L_{\{T=X\}} - L_{\{T=G(X)\}} = H(X) - H(G(X)) \tag{3.15}
$$

This essentially tells us that an intermediate representation with lower entropy could give a better initialization to the information bottleneck objective, which motivates us to transform real images into their simpler virtual counterparts.

Chapter 4

Experiments

4.1 Data

We use TORCS [35], an open-source car racing simulator, and Carla [36], a recent realistic urban driving simulator as our platform for virtual data collection. Figure 4.2 shows samples from both datasets. For TORCS, we construct a virtual dataset by setting up a robot car that follows a simple driving policy as defined in [9] and marking down its frontal camera images and steering commands. We also included twelve traffic cars that follow a simple control logic as defined in [9], with random noise added to the control commands to encourage varied behaviors. We captured our data on six game tracks with different shapes, which are summarized in Figure 4.1 To account for the imbalance of right turns and left turns in the virtual data, which could introduce bias in the domain transformation process, we augment our data by flipping the image and negate the steering command. For Carla, we use the training dataset provided by [37].

We use two large-scale real-world datasets released by Comma.ai [38] and Udacity [39] respectively (Table 4.1). Both datasets are composed of several episodes of driving videos. For the Comma.ai dataset, we follow the data reader provided by [38] and filter out data points where the steering wheel angle is greater than 200. For the Udacity dataset, we use the official release of training/testing data for challenge II in [39]. Large variance could be observed in lighting/road conditions and roadside views.

Dataset	train/test frames	Lighting	size
Commai.ai	345887/32018	Day/Night	160 x 320
Udacity	33808/5279	Day	240 x 320
Carla	657600/74600	Day/Dawn	88 x 200
TORCS	30183/3354	Day	240 x 320

Table 4.1: Dataset details.

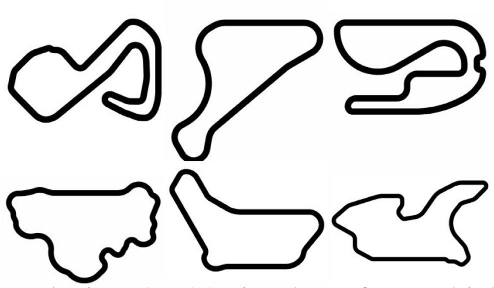


Figure 4.1: Shape of the 6 tracks in TORCS simulator from which virtual data are collected.

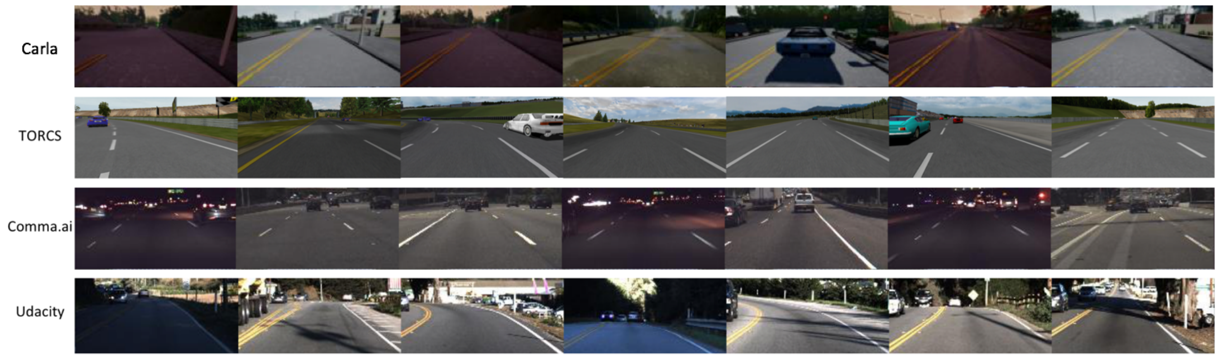


Figure 4.2: Sample data used by our work. From top to down: Carla (virtual), TORCS (virtual), Comma.ai (Real), Udacity (real)

4.2 Preprocessing

We first crop the input image to 160 x 320 by removing the extra upper part, which is usually background sky that does not change driving behavior. We then resize the image to 80 x 160 and normalize the pixel values to [-1, 1].

Instead of predicting the steering angle command directly, we predict the inverse of the radius as it is more stable and invariant to the geometry of the data-capturing car [6, 8]. The relationship between the inverse turning radius u_t and steering angle θ_t is characterized by the Ackermann steering geometry:

$$\theta_t = u_t d_w K_s (1 + K_{slip} v_t^2) \tag{4.1}$$

where θ_t is the steering command in radians, u_t(1/m) is the inverse of the turning radius, and v_t(m/s) is the vehicle speed at time t. d_w(m) stands for the wheelbase, which is the distance between the front and the rear wheel. K_{slip} is the slip coefficient. K_s is the steering ratio between the turn of the steering wheel and the turn of the wheels. We get d_w and K_s from car specifics released by the respective car manufacturer of the data-capturing vehicle, and use the K_{slip} provided by Comma.ai [38], which is estimated from real data. After predicting u_t, we transform it back to θ_t according to equation 4.1 and measure the mean absolute error of the steering angle prediction.

4.3 Training details

All the models are implemented in Tensorflow [40] and trained on an NVIDIA Titan-X GPU. We train all networks with the Adam optimizer [41] and set $\beta_1 = 0.5$. We follow the techniques used in [27] to stabilize the training. First, we use LSGAN [42], where the conventional GAN objective is replaced by a least square loss. Thus the loss function becomes

$$\begin{aligned}\mathcal{L}_d(D, G) =& \mathbb{E}_{\mathbf{x}^v}[D(\mathbf{x}^v; \theta_D)^2] + \qquad (4.2)\\ & \mathbb{E}_{\mathbf{x}^r}[(1 - D(G(\mathbf{x}^r; \theta_G); \theta_D))^2], \qquad (4.3)\end{aligned}$$

Second, we train the discriminator using a buffer of generated images to alleviate model oscillation [31]. We use a buffer size of 50.

In order to take advantage of the labeled data collected from simulators, we initialize the predictor network with a model that is pretrained on virtual images. During pretraining, we set the batch size to 2000 and learning rate to 0.01.

At each step, we sequentially update θ_G, θ_P and θ_D with respect to the objective functions in 3.5, 3.6 and 3.7. We use a batch size of 60. We set $\alpha_P = 0.0002$, $\alpha_{GAN} = 0.00002$, and $\lambda = 0.5$ to 1. We train the model for a total of 7 epochs.

After obtaining a real-to-virtual generator for each real domain, we could fix the generator and train a global predictor with all real datasets. We initialize the global predictor with *PilotNet* pretrained on virtual data, and use a learning rate of 0.001 and a batch size of 2000 for training.

4.4 Metrics and baselines

We evaluate the effectiveness of our model in terms of the quality of generated images in the virtual domain and the mean absolute error of steering angle prediction. We compare the performance of DU-drive with the following baselines. To ensure fairness, we use the same architecture for the predictor network as described in section 3.1.

- **Vanilla end-to-end model (*PilotNet*)** Our first baseline is a vanilla end-to-end model, where a real driving image is directly mapped to a steering command. This is essentially the same model (*PilotNet*) as employed by [6].
- **Fine-tune from virtual data** A straightforward way to do domain adaption is to fine-tune a model pretrained in the source domain with data in the target domain. We first train a predictor with virtual data only, then fine-tune it with the real dataset.
- **Conditional GAN** A naive implementation of conditional GAN (cGAN) [34] uses a generator G to transform an image x from the real domain to an image $G(x)$ in the virtual domain. A discriminative network D is set up to discriminate $G(x)$ from y sampled from the virtual domain while G tries to fool the discriminator. No additional supervision is provided other than the adversarial objective. We also train a *PilotNet* to predict steering angle from the fake virtual image generated by cGAN.
- **CycleGAN** CycleGAN [28] is a method to do unpaired image-to-image translation, which could also be applied to transforming real driving images to the virtual domain. It uses one generative adversarial network G_{X2Y} to transform image x from domain X to image

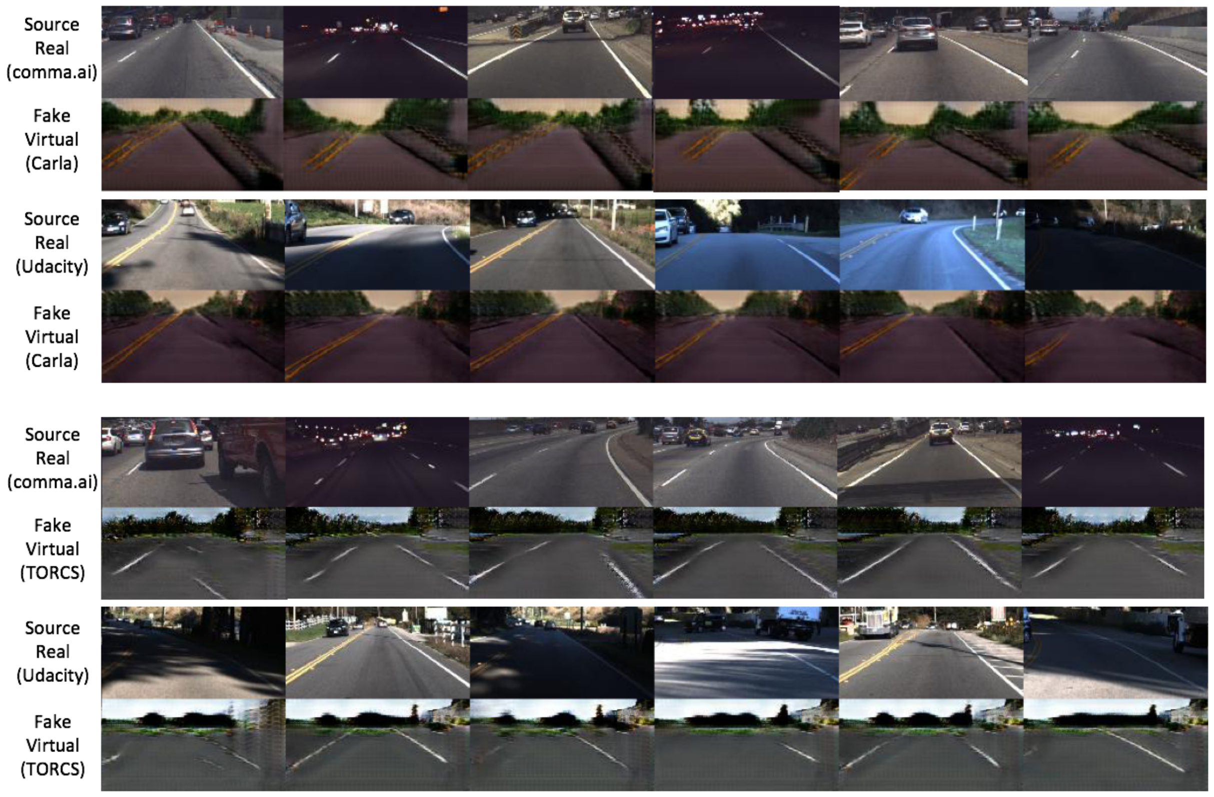


Figure 4.3: Image generation results of DU-Drive. Information not critical to driving behavior, e.g. day/night lighting condition and the view beyond the road boundary, is unified. Driving-critical cues like lane markings are well preserved.

$G_{X2Y}(x)$ in domain Y, and another generative adversarial network G_{Y2X} to transform $G_{X2Y}(x)$ back to an image $G_{Y2X}(G_{X2Y}(x))$ in domain X. A cycle consistency loss is employed to ensure that $G_{Y2X}(G_{X2Y}(x)) = x$.

- ***PilotNet* joint training** To verify the effectiveness of our domain unification model, we also directly train a *PilotNet* with two labeled real datasets simultaneously.

4.5 Quantitative Results and Comparisons

We compare the performance of steering command prediction for a single real domain of our DU-drive (single) model with the plain end-to-end model (*PilotNet*), fine-tuning from virtual data and conditional GAN without joint training (Table 4.2). Both DU-drive (single) and fine-tuning from virtual data perform better than the plain end-to-end model, which verifies the effectiveness of leveraging annotated virtual data. DU-drive (single) outperforms fine-tuning by 12%/20% using TORCS virtual data and 11%/41% using Carla virtual data for the Comma.ai/Udacity datasets respectively, despite using the same training data and prediction network. This verifies the superiority of transforming complex real images into their simpler counterparts in the virtual domain for the driving command prediction task. Conditional GAN without joint training does not perform well as the adversarial objective itself is not enough to ensure the preservation of label.

Simulator		TORCS		Carla	
Dataset	Model	MAE	SD	MAE	SD
Udacity	*PilotNet*[6]	6.018	7.613	6.018	7.613
	fine-tune from virtual	5.808	7.721	6.053	8.041
	cGAN [34]	5.921	6.896	4.925	7.100
	PilotNet joint training	15.040	27.636	15.040	27.636
	DU-Drive (single)	4.558	**5.356**	**3.571**	4.958
	DU-Drive (unified)	**4.521**	5.558	3.808	**4.650**
Comma.ai	*PilotNet*[6]	1.208	1.472	1.208	1.472
	fine-tune from virtual	1.203	1.500	1.196	1.473
	cGAN [34]	1.215	1.405	1.206	1.404
	PilotNet joint training	5.988	11.670	5.988	11.670
	DU-Drive (single)	**1.061**	1.319	**1.068**	**1.337**
	DU-Drive (unified)	1.079	**1.270**	1.174	1.460

Table 4.2: Mean absolute error (MAE) and standard deviation (SD) for the steering angle prediction task. DU-drive clearly outperforms all baseline methods.

4.6 Information Bottleneck Analysis of Virtual Representation

As shown in Table 4.2, transforming real images to the virtual domain using our DU-drive model gives superior performance even with the same training data and predictor network. We attribute this to the fact that virtual images are more homogeneous and contains less complexity that is not related to the prediction task. As shown in Figure 4.3, superfluous details including views beyond the road and changeable lighting conditions are unified into a clean, homogeneous background, while cues critical for steering angle prediction like lane markings are preserved. In the language of information bottleneck theory, this corresponds to a representation that is closer to the optimal minimum sufficient statistic than the raw image with respect to the prediction task.

Following the deduction in 3.4, we now show by estimation that $H(X) > H(X_v)$, which implies $L_{\{T=X\}} > L_{\{T=X_v\}}$. While it is unclear how to measure the entropy of an arbitrary set of images, under the mild assumption of normal distribution, the entropy equals the natural logarithm of the determinant of the covariance matrix up to a constant. We therefore treat each image as a vector and measure the total variance of 50 randomly sampled pairs of real and generated virtual data. As shown in the results summarized in Table 4.3, virtual representation tends to have lower entropy and hence gives a better initialization to the information bottleneck objective. Figure 4.4 shows that for the four data points we obtained from our experiments, the

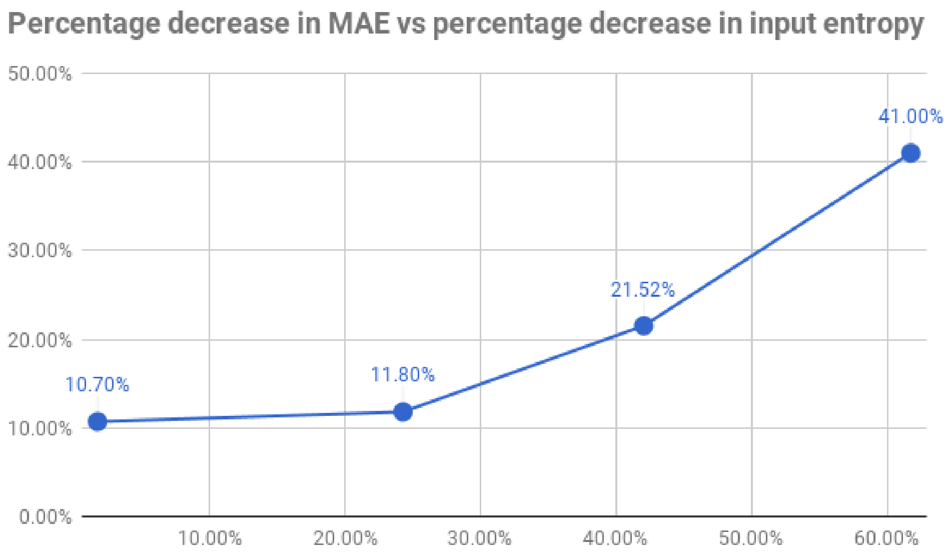


Figure 4.4: The percentage decrease in MAE versus the percentage decrease in input entropy. The four data points are from the combination of two real datasets and two simulators. The percentage decrease in MAE is calculated between fine-tune and DU-drive (single).

percentage decrease in prediction mean absolute value is positively corelated with the percentage decrease in input entropy.

Variance	Carla		TORCS	
	Udacity	Commaai	Udacity	Commaai
Real	82745	23902	107666	29656
Virtual	31650	23483	62389	22453

Table 4.3: Variance of 50 randomly sampled pairs of real and generated virtual images. The generated virtual images have lower variance, which implies lower entropy for input distribution and thus less burden during the compression phase when optimizing the information bottleneck tradeoff.

4.7 Effectiveness of Domain Unification

A critical advantage of our model is that data collected from different sources can be unified in the same virtual domain. As shown in Figure 4.3, images from the Comma.ai dataset and those from the Udacity dataset are transformed into a unified virtual domain, whose superiority is directly reflected in the performance of the steering angle prediction task. As shown in Table 4.2, directly training a network with data from two real domains together will lead to results much worse than training each one separately due to domain shift. However, with DU-drive (unified), a single network could process data from multiple real domains with comparable results to DU-drive (single). Moreover, DU-drive separates the transformation and prediction process, and a generator could be independently trained for a new real dataset.

To further study the generalization ability of DU-drive, we conducted semi-supervised experiments where labels are limited for an unseen dataset. We first train a DU-drive model with the Comm.ai data, then use 20%/50% of the labeled Udacity data respectively to train the generator with our co-training scheme and report the prediction performance on the test set. We also experimented on joint training with the Comma.ai dataset under our domain unification framework. As shown in Table 4.4, Domain unification outperforms baselines by a large margin, especially when labeled data are scarce. This shows the superiority of domain unification at transferring knowledge across domains and alleviating domain shift.

Percentage	Carla			TORCS		
	PilotNet	Ours(single)	Ours(unified)	PilotNet	Ours(single)	Ours(unified)
20%	7.86	7.12	**6.02**	7.86	6.85	**6.34**
50%	7.11	6.41	**5.15**	7.11	5.73	**5.42**
100%	6.02	**3.57**	3.81	6.02	4.56	**4.52**

Table 4.4: MAE for semi-supervised learning.

4.8 Comparison with CycleGAN

Although CycleGAN seems to do better at recovering details (Figure 4.5), which could be important in many generation tasks, the additional details that are unrelated to steering angle prediction do not help much for our task. They also come at a cost of high memory usage, which could make joint training with the predictor difficult. Moreover, since real images are more complex than virtual images, the generated fake real images often include lots of undesired hallucinated details, which justifies our motivation to go the other way around.

4.9 Prevention of mode collapse

Mode collapse is a common pitfall for generative adversarial networks. Due to the lack of additional supervision, a naive implementation of conditional GAN easily suffers from unstable training and mode collapse (Figure 4.6). With our novel joint training of steering angle prediction and real-to-virtual transformation, mode collapse for driving-critical information like lane markings is effectively prevented.

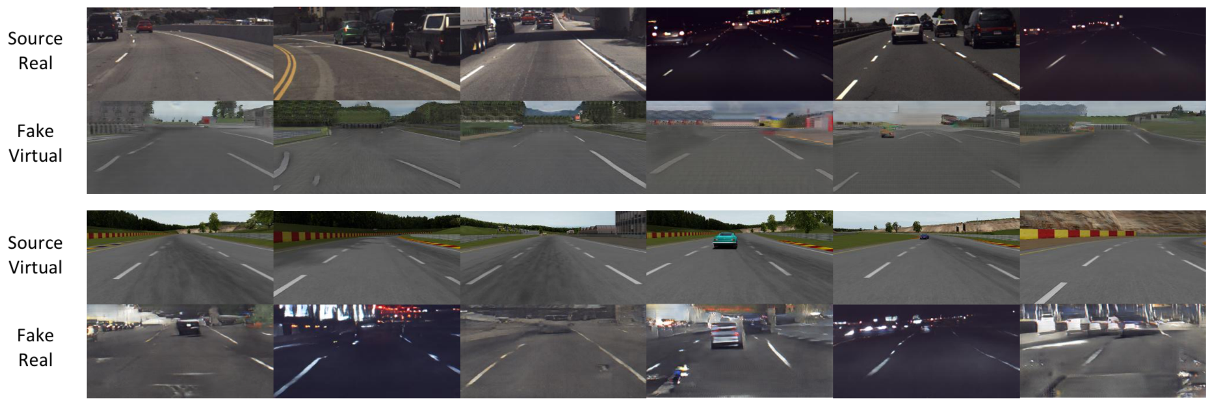


Figure 4.5: Image generation results of CycleGAN. Top row: real source images and generated fake virtual images. Bottom row: virtual source images and generated fake real images.

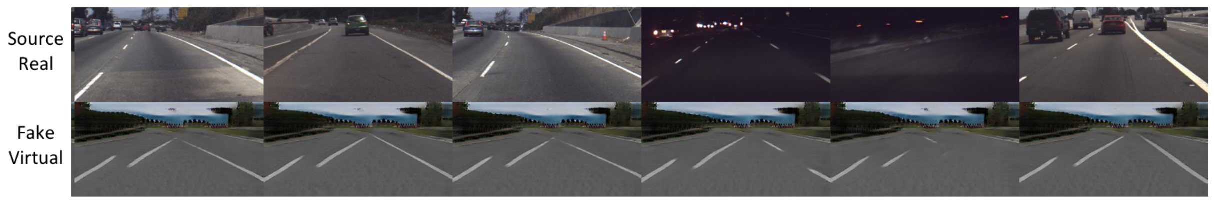


Figure 4.6: Mode Collapse happens for naively implemented conditional GAN.

Chapter 5

Conclusion and Future Work

We propose a real-to-virtual domain unification framework for autonomous driving, or DU-drive, that employs a conditional generative adversarial network to transform real driving images to their simpler counterparts in the virtual domain, from which the vehicle control commands are predicted. In the case where there are multiple real datasets, a real-to-virtual generator was able to be independently trained for each real domain and a global predictor was able to be trained with data from multiple sources simultaneously. We further analyze the advantage of using the virtual representation under the framework of the information bottleneck theorem. Qualitative and quantitative experiment results clearly show that our model can effectively unify real images from different sources to more efficient representations in the virtual domain, eliminating domain shift and boost the performance of the control command prediction task.

One important direction for future work is the deployment of our method in dynamic environments. For sequential prediction tasks such as vehicle control, good supervised-learning testing results on a given dataset don't guarantee the successful execution of the task when interacting with a dynamic environment due to the accumulation of error. To this end, many interactive algorithms such as DAgger [43] and reinforcement learning have been developed to train policies that could successfully maneuver the robot in a dynamic environment. It should be straightforward to combine our method with such interactive learning algorithms by simply replacing the perception module with our real-to-virtual transformation framework.

Run-time performance is another issue to be considered. In our experiments carried out on a single Titan-X GPU, the average processing time for one frame of image is 0.0112s, which should be sufficient for real-time applications. However, vehicle-mounted GPUs tend to be less powerful than cluster GPUs and the computation power needs to be shared among many different tasks in the self-driving stack. It is therefore important to integrate our framework with other parts of the self-driving stack and optimize the online performance.

Bibliography

[1] Lucy Reading-Ikkanda/Quanta Magazine. Inside deep learning, 2017. URL `https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921/`. Online; accessed April 23, 2018. (document), 2.1

[2] Charles Thorpe, Martial H Hebert, Takeo Kanade, and Steven A Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988. 1, 2.1

[3] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 1, 2.1

[4] Ernst D Dickmanns, Birger Mysliwetz, and Thomas Christians. An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1273–1284, 1990. 1, 2.1

[5] Ernst Dieter Dickmanns and Volker Graefe. Dynamic monocular machine vision. *Machine vision and applications*, 1(4):223–240, 1988. 1, 2.1

[6] 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. 1, 2.1, 3.1, 4.2, 4.4, **??**, **??**

[7] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182, 2017. 1, 2.1

[8] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2.1, 4.2

[9] 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. 1, 2.1, 4.1

[10] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017. 1, 3.1

[11] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle.

In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015. 1, 2.3

[12] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017. 1, 3.4

[13] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011. 1, 2.4

[14] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017. 2.1

[15] Autonomous off-road vehicle control using end-to-end learning. `http://net-scale.com/doc/net-scale-dave-report.pdf`. Accessed: 2017-10-20. 2.1

[16] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015. 2.1

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2.2

[18] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821. IEEE, 2017. 2.2

[19] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2.2

[20] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 2.3

[21] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM, 2007. 2.4

[22] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015. 2.4

[23] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 7167–7176. IEEE, 2017. 2.4

[24] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015. 2.4

[25] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015. 2.4

[26] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016. 2.4

[27] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2.4, 4.3

[28] 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. 2.4, 3.1, 4.4

[29] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 2.4, 3.1

[30] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017. 2.4, 3.2

[31] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2.4, 3.2, 4.3

[32] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017. 2.4

[33] Yurong You, Xinlei Pan, Ziyan Wang, and Cewu Lu. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017. 2.4

[34] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 3.1, 4.4, **??**, **??**

[35] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 2000. 4.1

[36] 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. 4.1

[37] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *International Conference on Robotics and Automation (ICRA)*, 2018. 4.1

[38] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016. 4.1, 4.2

[39] Udacity self-driving-car challenge dataset. `https://github.com/udacity/self-driving-car/tree/master/datasets`. Accessed: 2017-10-20. 4.1

[40] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,

Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 4.3

[41] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 4.3

[42] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 4.3

[43] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 5

[44] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 2108–2116. IEEE, 2017.

[45] 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.