UNIVERSITY OF CALIFORNIA
Merced

Improving Semantic Segmentation for Autonomous Vehicles using Synthetic Images

A Master Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

by

Mehul Divecha

Committee in Charge:

Professor Shawn Newsam, Chair

Professor Ming-Hsuan Yang

Professor Mukesh Singhal

May 2019

Improving Semantic Segmentation for Autonomous Vehicles using Synthetic Images

by

Mehul Divecha

The Master Thesis of Mehul Divecha is approved and it is acceptable in quality and form for publication on microfilm and electronically:

Professor Mukesh Singhal

Professor Ming-Hsuan Yang

Professor Shawn Newsam, Chair

University of California, Merced

2019

To my family and friends for their endless

support and patience.

Acknowledgements

I would like to thank my advisor Prof. Shawn Newsam for the continuous support during my Master's and related research for his guidance, patience and knowledge. His invaluable advice helped me immensely during all the steps of my research, the writing of my thesis as well other facets of my academic career.

I would like to thank the rest of my thesis committee: Prof. Ming-Hsuan Yang and Prof. Mukesh Singhal, for their comments and opinions.

I would like to thank my family and friends for their never ending support and immense patience throughout writing this thesis and my education.

Abstract

Improving Semantic Segmentation for Autonomous Vehicles using Synthetic Images

Mehul Divecha

With the prevalence of Advanced Driver's Assistance Systems (ADAS) and a surge in interest in autonomous vehicles, it has become important that the computer vision modules that make up these systems understand their natural surroundings and react appropriately to changes. A key aspect to understanding such natural scenes is to identify the locations and bounds of the objects present in the scene. Semantic segmentation is one to way to approach this problem. With the rise of deep learning techniques, there has been a tremendous progress in semantic segmentation with great improvements in quality and performance. However, one down-side of most deep learning methods is the requirement of a large set of annotated data. This becomes very cumbersome when it comes to segmentation problems, since they require pixel level annotations. Another issue that arises is that of a domain gap introduced when deploying a model on data that is different from what it was trained on. In this thesis we tackle the first issue by leveraging a practically unlimited source of annotated in the form of game engines and virtual environments. We then transform the data thus derived

to have a more photo-realistic look matching their real-world counterparts, thus aiming to solve the second issue. We describe the process we have employed to transform the synthetic looking images to look as close to the real-world images as possible and show that there are significant gains to be had by adopting such a method.

Contents

List of Figures

List of Tables

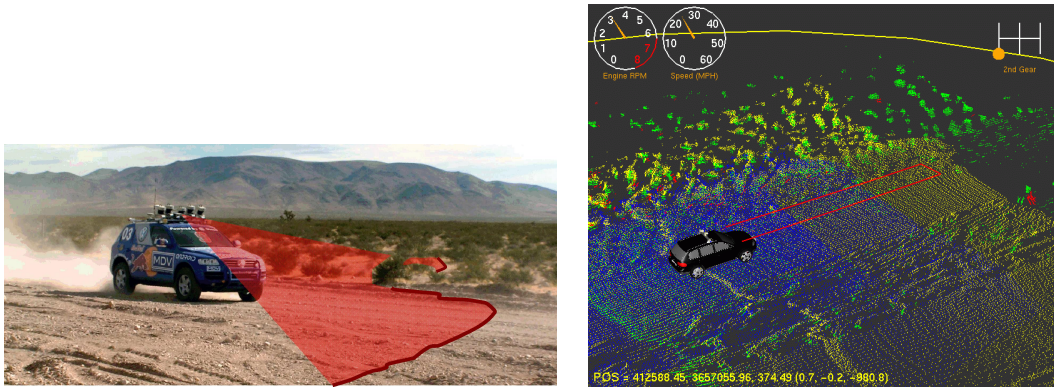
Chapter 1

Introduction

With the advent of autonomous vehicles, the need for understanding natural scenes has become quite critical. Autonomous vehicles, particularly self-driving cars, need to be able understand and recognize their immediate surroundings, both with very high accuracy as well as in a timely manner.

An autonomous car drives in a world that is very complex and dynamic. It not only has a short time to react to the changes in the environment, but it needs to be very accurate in identifying the changes. Compromise in either its ability to react in time or correctly recognize its surrounding can lead to life-threatening situations.

Up until a few years ago, it was deemed impossible to build a self-driving car that relied solely on vision. Hence, approaches that complemented vision with



(a) (b)

Figure 1.1: (a) Stanley possesses five laser sensors mounted at five different angles and angled downwards. These sensors scan the terrain in front of the vehicle as it moves. (b) Each of the laser sensor acquires a point cloud across multiple scans and over time. This point cloud is then analyzed for drivable terrain and potential obstacles.

other modalities were tried and found to be successful. Stanley [29] was one of the first successful self-driving cars that heavily relied on LIDAR for its navigation. LIDAR works by mapping the environment by measuring the reflected pulses of a laser. This technique generates a point cloud that gives a depth estimate of surrounding objects. An illustration of this process in Stanley is shown in Fig 1.1. However, it's not without shortcomings. Apart from the high costs of deploying and maintaining the hardware, it's also expensive to analyze point clouds. Additionally, reflections, rain and snow make LIDAR data too noisy to

be effectively analyzed. Also, LIDAR data is sparse when compared to the dense data that cameras generate.

In recent years, with the advent of deep learning, it has become quite feasible to accurately and effectively process large amount of visual data. Deep learning has shown to outperform classical techniques in many areas like image classification, object detection, object recognition and semantic segmentation [16], [10], [23], [22], [4], [9]. Natural scene understanding relies heavily on semantic segmentation to achieve its goal. The goal of semantic segmentation is to classify each pixel in an image and label it with a category. In a way, semantic segmentation can be thought of as a dense classification problem applied to each pixel of an image instead of the entire image at once.

Although deep learning outperforms traditional methods in many applications, one requirement for its success is that it needs a large amount of training data, an order of magnitude higher than what was traditionally needed. This is because models in deep learning have higher model complexity. This needs a large amount of data to act as regularizer [1]. For supervised learning problems like classification or segmentation, this means that this large amount of data needs to be annotated. This becomes a very labor intensive task for semantic segmentation as it needs ground truth annotations for every pixel in the image, for all such images in a dataset.

In this work, we adopt a methodology of training a segmentation model on synthetic data and then testing its performance on real world data. There often exists a domain gap between the training and test dataset. This occurs because the synthetic data have a significant difference in their visual appearance than the real world. To achieve optimum performance, the domain gap needs to be as small as possible. One way to achieve that is by transforming the synthetic data to have more photorealism and look more like the real world data.

In this work, we utilize some recent advances in Generative Adversarial Networks to transform synthetic images to be photorealistic, and then using these as training data for segmentation models. The idea here is to make use of the practically infinite amount of synthetic data available to us instead of the finite and hard to obtain real world data. Once we have obtained photorealistic training data, where the domain gap between train and test data is as small as possible, we train segmentation models using this transformed data and its annotations only. This saves us the need of compiling annotated real world training data.

Synthetic data for computer vision problems is primarily obtained by modifying games or building simulators that generate annotations readily. At the heart of these are game engines, like Unity [30] and Unreal [6] that render the graphics and provide object placement and depth information. A variety of ground truth data can be generated. For example, CARLA [5] can generate segmentation and

depth maps and virtual LIDAR data, apart from vehicle telemetry and virtual GPS. For image segmentation, we are primarily interested in segmentation maps that associate each pixel with a class or an instance of a class.

The rest of the thesis is organized as follows. In chapter 2, we discuss datasets used for evaluations, wherein we describe datasets gathered in real world and annotated manually as well as datasets generated in virtual environments with automatic annotations. Chapter 3 discusses the relevant models utilized in this work. Section 3.1 talks about the primary segmentation model, DeepLab-v3+, briefly discussing its architecture and some of its components that are relevant to this work. Section 3.2 discusses UNIT, the framework we use to transform synthetic image data to look more photorealistic. The section also gives a brief overview of some of the recent developments in generative modeling such as Variational Autoencoders and Generative Adversarial Networks. Chapter 4 discusses the methodology and the results of this work and finally we conclude with a summary and future work in Chapter 5.

Chapter 2

Datasets for Urban Scene Understanding

2.1 SYNTHIA dataset

SYNTHIA (SYNTHetic collection of Imagery and Annotations) has been generated to aid semantic segmentation with a specific emphasis on autonomous driving problems. Although specifically targeted towards semantic segmentation, it is also useful for other ADAS (Advanced Driver's Assistance System) and autonomous driving tasks, like object recognition, place identification and change detection.

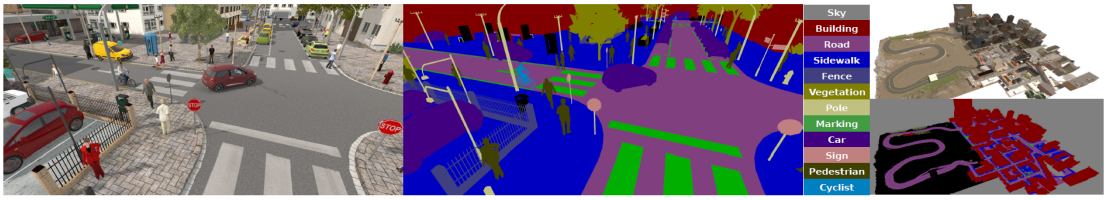


Figure 2.1: Sample of images from SYNTHIA dataset

The dataset contains images that are individual frames rendered from a virtual city. The labels are pixel level semantic annotations for 13 classes in all. Fig 2.1 shows the 13 classes, which are: sky, building, road, sidewalk, fence, vegetation, lane-marking, pole, car, traffic signs, pedestrians, cyclists and miscellaneous. Each of the frame has an associated depth map. The frames are acquired from different locations and viewpoints, with up to eight viewpoints per location.

2.1.1 Virtual World Generator

The frames for SYNTHIA have been rendered in a virtual urban environment that has been created using the Unity game engine [30]. The environment has been modeled after typical cities that includes some of the most common and frequently observed elements like people, shops, parks and gardens, vegetation, streets and blocks, lane markings, traffic signs, pavements, lamp poles, highways and rural areas. These elements form the basic building blocks for the virtual environment that can be combined in any different manner as suited to create

new scenes, and the annotations for each are obtained at no additional labor or computational expense. Even the basic properties of each element, such as color, shape and texture, can be individually controlled to create unique looks and improve variability of the data.

SYNTHIA attempts to achieve photorealism by a variety of means to get the virtual environment to look as real as possible. Firstly, the virtual world can be modelled to have four different seasons for varied appearances, with snowy winters, sunny summers, wet rainy season and flowery spring. A dynamic illumination engine can produce different illumination conditions, which models different times of the day, like noon and dusk, in addition to days that are sunny and cloudy. Additional realism is added by the engine by simulating realistic shadows cast by the clouds and other objects in the scene

2.1.2 SYNTHIA-Rand and SYNTHIA-Seqs

SYNTHIA comprises of two datasets: SYNTHIA-Rand and SYNTHIA-Seqs. The images have a resolution of 960×720 pixels and a horizontal field of view of $100°$. SYNTHIA-Rand consists of 13400 frames and was generated by an array of virtual cameras randomly spread across the city. Several frames correspond to a single camera location, each with different elements in the scene, textures for each element and illumination of the scene.

Figure 2.2: Sample from Cityscapes dataset

SYNTHIA-Seqs consists of four video sequences obtained from driving around a virtual car in the environment across different seasons. Each sequence has about 50,000 frames, thus comprising of 200,000 frames in total. The virtual platform for acquisition comprises of 8 cameras mounted on various locations of the car with overlapping field of view to create a 360° view of the scene. Each of the cameras also has an associated depth sensor that is aligned with the camera center and has a range of $1.5m$ to $50m$. A dynamic behavior for the virtual vehicle is created when it interacts with other dynamic elements of the environment like pedestrians and other vehicles.

2.2 Cityscapes dataset

Cityscapes is a large-scale dataset captured in real world setting, with the goal of providing a rich set of data for semantic segmentation and scene understanding.

2.2.1 Data specifications

Cityscapes comprises of several hundreds of thousands of frames acquired from a moving vehicle spanning 50 cities in Germany as well as neighboring countries. The duration of acquisition ranges over several months and encompasses the seasons of spring, summer and fall. The dataset lacks representation for adverse weather conditions such as heavy rain or snow, as the authors believe that would require specialized techniques and datasets.

The recording of the images was done using an automotive grade stereo camera at a frame rate of 17 Hz. With the sensors mounted behind the windshield, the images are high dynamic range with 16 bits color depth. Furthermore, these images were converted to 8-bit low dynamic range by applying a logarithmic compression curve, to provide comparability and compatibility with existing datasets. 5000 images were chosen from 27 cities for dense pixel level annotation, with the aim of providing high diversity of foreground objects, background and overall scene layout. Every 20th frame of a 30 frame video snippet was selected for such annotation. Another 20,000 images in total were obtained from the remaining 27 cities by coarsely annotating an image for every 20m of driving distance. Apart from the images and their corresponding annotations, other information provided by the dataset includes vehicle odometry from in-vehicle sensors, outside temperature and GPS tracks.

2.3 GTA5 dataset

While SYNTHIA attempts to generate training data by creating a virtual environment it can compose and control, it falls short when it comes to photorealism. The lack of photorealism increases the domain gap between the training and the target datasets. [25] attempt to solve the issue by leveraging the photorealistic environments present in commercial games, particularly Grand Theft Auto 5, abbreviated as GTA5. GTA5 [7] is an open world simulation game with a variety of driving scenarios where the rendering engine creates convincing and photorealistic looking frames. Figure 2.3 shows a sample from the dataset created, henceforth known as GTA5 dataset. Below we briefly describe the acquisition and annotation procedure adopted in [25] to create the dataset.

2.3.1 Data acquisition

Since GTA5 is a commercial game and not open-source, it is not easy to access its internal game engine content. However, there are ways in which it is possible to intercept the communication between the game and GPU and make use of this information in constructing a dataset. Particularly, the authors use a technique called detouring [11] wherein they create a wrapper around the system graphics API and intercept the calls being made by the game. A wrapper implementing

(a) Images (b) Segmentation

Figure 2.3: Sample of images in GTA5 dataset

such detouring is RenderDoc [12] which the authors utilize to access the game data. For every 40th frame in the game, the authors intercept the calls made to

graphics subsystem and collect all the resources that is relevant for annotation and to reproduce the frames, particularly the meshes, textures and shaders used internally to render it. The resources associated with the mesh, texture and shader are hashed to create a 128-bit key of their memory locations. These hashes are used to uniquely identify the usage of these resources across different frames. To associate these resources to each individual pixel in the frame, a two pass approach is taken: 1) In the first pass, a color image and its associated buffers are created, similar to how the frame is rendered in the game, 2) and in the second pass each pixel then stores the resource ID of mesh, texture and shader that designate the scene element at that pixel.

The images thus obtained are then decomposed into patches that share a common mesh, texture and shader combination. These patches resemble superpixels, though they have some advantages over superpixels: 1) The patches are associated with the underlying surface in the scene and such surfaces can be easily linked across different images, 2) the patches are edge accurate, that is, their boundaries coincide with that of the semantic class and the instance of that class exactly, and 3) the metadata associated with each patch can be used to propagate labels across instances of object that may not share the same mesh, texture and shader combination. To speed up the annotation process, associative rule mining is performed to identify statistical regularities between a resource and a semantic class. The

final step in the annotation process is manual labeling the patches and grouping the patches that belong to a single instance together. This is done via an interactive interface created by the authors. The first frame takes longer to annotate since the patches are very granular, however, once it is done, the annotation tool propagates these labels to the subsequent frame.

Chapter 3

Models for Semantic Segmentation

3.1 DeepLab-v3+

In this section we discuss the DeepLab-v3+ architecture that forms the basis of our evaluation. DeepLab-v3+ [4] builds upon previous similar architectures, also called DeepLab ([2], [3]). It is based on a family of architectures called encoder-decoder architectures.

As shown in Figure 3.1, a decoder module is coupled with the encoder module to recover detailed object boundaries. The output of DeepLab-v3+ encodes rich

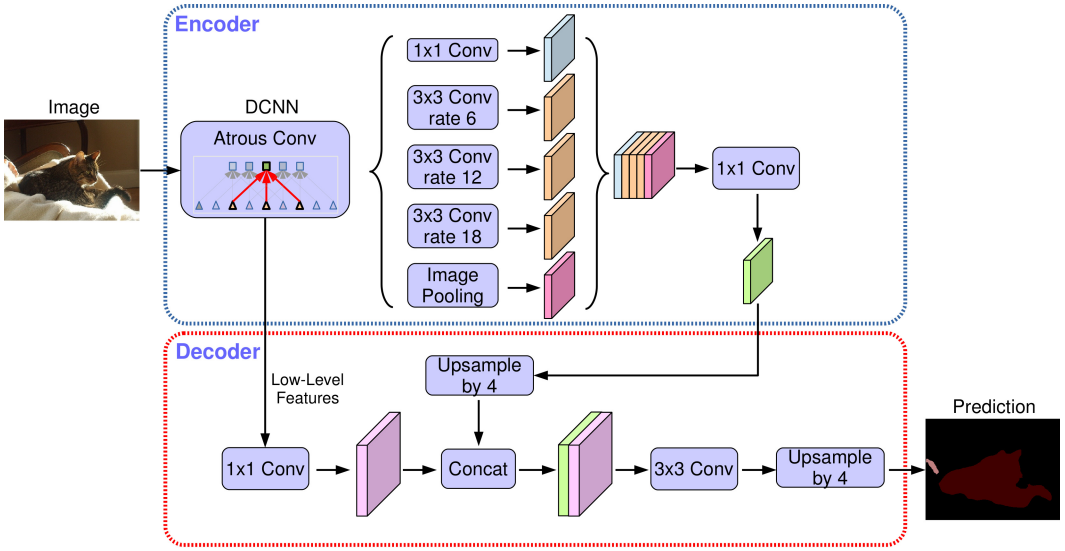
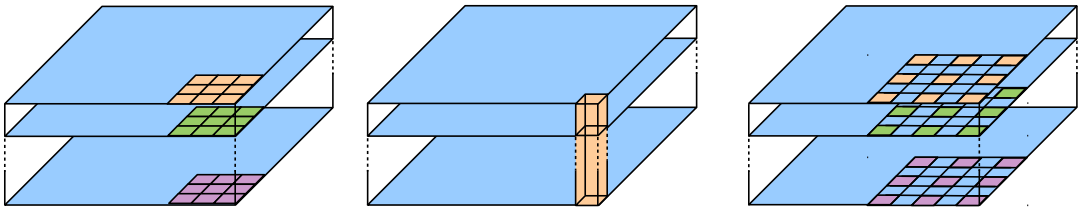


Figure 3.1: Representation of the DeepLab-v3+ architecture. The encoder applies atrous convolution at multiple scales to encode multiscale contextual information. The decoder is a simple yet effective module that refines the segmentation results along object boundaries.

semantic information, which the atrous convolution allows us to extract encoder features from, at various scales.

Atrous convolution generalizes standard convolution operation and allows us to adjust the filters' field-of-view in order to capture multiscale information. For an input feature map x, output feature map y, convolution filter w and for each location i in y, atrous convolution is applied as follows:

$$y[i] = \sum_{k} x[i + r \cdot k]w[k]$$



(a) Depthwise conv. (b) Pointwise conv. (c) Atrous depthwise conv.

Figure 3.2: A 3×3 depthwise separable convolution is composed of a (a) a depthwise convolution, which applies a single filter for each input channel and (b) a pointwise convolution, which combines the outputs from depthwise convolution across channels. The authors discuss *atrous separable convolution* where atrous convolution is adopted in the depthwise convolution. (c) shows atrous separable convolution with $rate = 2$.

where r is the stride with which the input is sampled. Standard convolution is a special case in which $r = 1$. By changing r, we can adaptively modify the filters' field-of-view.

To reduce computational complexity, DeepLab-v3+ uses depthwise separable convolution. Depthwise separable convolution factorizes a standard convolution into a depthwise convolution followed by a pointwise convolution, that is, it first applies spatial convolution to each channel independently and then performs a pointwise convolution to combine these outputs. Figure 3.2 illustrates this process. DeepLab-v3+ uses atrous convolution for depthwise convolution and the resulting convolution is referred to as atrous separable convolution. Atrous separable

convolution have been shown [3] to significantly reduce computational complexity without sacrificing performance.

In the DeepLab-v3+ encoder, atrous convolution is very effective in extracting features at multiple resolutions. If we refer to the ratio of input resolution to the final output resolution as output stride, then for the task of image classification, the stride is usually 32, as the final maps feature maps are often 32 times smaller than the input resolution. For the task of semantic segmentation, since we need denser feature extraction, we adopt an output stride of either 16 or 8. This can be done removing the striding in the last blocks and applying the atrous convolution correspondingly. In addition, DeepLab-v3+ augments the Atrous Spatial Pyramid Pooling module, which probes convolutional features at multiple scales by applying atrous convolution, with the image-level features [20]. The penultimate layer in the encoder (the one before the logits) is used as the encoder output. It consists of 256 channels and is rich in semantic information. Moreover, depending on the computational budget, features can be extracted at arbitrary resolution by applying the atrous convolution.

The encoder features from DeepLab-v3+ are usually computed with an output stride of 16. In [3], a very simple decoder module is constructed by bilinearly upsampling the features by a factor of 16. This naive decoder module fails to successfully recover segmentation details for the objects. DeepLab-v3+ introduces

a simple and effective decoder module, as shown in Fig 3.1. It works by first bilinearly upsampling the encoder features by a factor of 4 and then concatenating them with the corresponding low level features of the same spatial resolution from the backbone network. Since the low level features often contain a large number of channels (e.g., 256 or 512), this may make training harder by outweighing the importance of the semantically rich encoder features. This has been mitigated in the decoder by 1×1 convolution on the low level features to reduce the number of channels. Once the above mentioned concatenation is done, 3×3 convolutions are applied to refine the features followed by one more bilinear upsamplings by a factor of 4. The authors demonstrate that using an output stride of 16 for the encoder module provides a good balance between speed and accuracy.

3.2 Unsupervised Image-to-image Translation

Image-to-image translation is a class of problems where in images from one domain are "translated" to look like images from another domain. Although it is easier in the supervised regime, wherein a pair of images from the two domains are chosen manually, the problem of image-to-image translation becomes difficult when we need to transform distributions in an unsupervised manner. If we consider X_1 and X_2 to be image domains, then in supervised image-to-image trans-

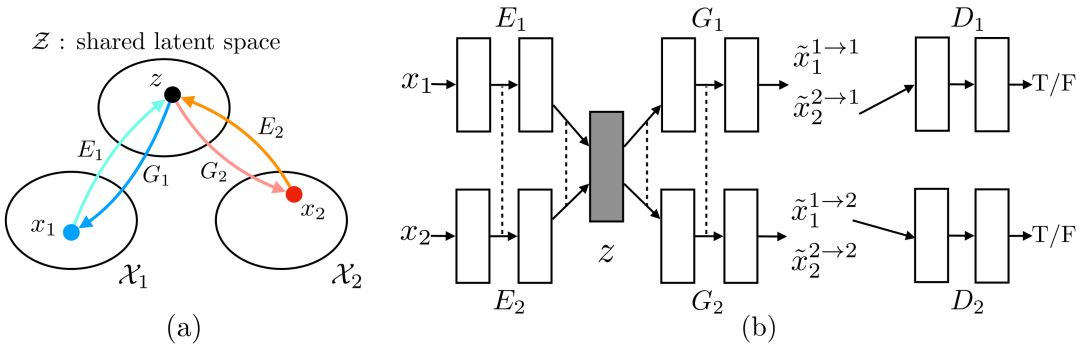


Figure 3.3: (a) Illustration of the shared latent space assumption. It is assumed that a pair of corresponding images (x_1, x_2) in two different domains $\mathcal{X}_1$ and $\mathcal{X}_2$ can be mapped to a same latent code z in a shared latent space $\mathcal{Z}$. Here, the images are mapped to latent codes by two encoding functions E_1 and E_2. The latent codes are mapped to images by two generation functions G_1 and G_2. (b) Illustration of UNIT framework. To implement the shared latent assumption, E_1, E_2, G_1 and G_2 are represented using CNNs and a weight sharing constraint is employed where the connection weights of the high level layers (last few layers) of E_1 and E_2 are tied, as illustrated by dashed lines. Similarly the connection weights of the high level layers (first few layers) of G_1 and G_2 are tied. D_1 and D_2 are adversarial discriminators for the respective domains, in charge of evaluating whether the translated images are realistic. $\tilde{x}_1^{1\to1}$ and $\tilde{x}_2^{2\to2}$ represent the self-reconstructed images and $\tilde{x}_1^{1\to2}$ and $\tilde{x}_2^{2\to1}$ represent the domain translated images. (Figures adapted from [18].)

lation, a pair of samples (x_1, x_2) is drawn from a joint distribution $P_{X_1,X_2}(x_1, x_2)$. However, in an unsupervised setting, the samples pair is drawn from the marginal

distributions $P_{X_1}(x_1)$ and $P_{X_2}(x_2)$. The inference of the joint distribution from the marginals is intractable in this case without additional assumptions.

To alleviate this problem, UNsupervised Image-to-image Translation (UNIT) [18] makes the assumption of shared latent space. Figure 3.3 shows a code z lying in the shared latent space associated with a pair of samples x_1 and x_2. The shared latent space is constructed in a way that makes it feasible to recover both the images from z and, inversely, z can be computed from each of the two images. To construct this shared latent space code, UNIT postulates there exist functions E_1^*, E_2^*, G_1^*, and G_2^* such that, for a given pair of corresponding images (x_1, x_2) from the joint distribution, we have $z = E_1^*(x_1) = E_2^*(x_2)$ and conversely $x_1 = G_1^*(z)$ and $x_2 = G_2^*(z)$. To map from X_1 to X_2, the model defines a function $F_{1\to2}^*$ such that $x_2 = F_{1\to2}^*(x_1)$, where $F_{1\to2}^*$ is a composition of G_2^* and E_1^*, defined as $F_{1\to2}^*(x_1) = G_2^*(E_1^*(x_1))$. Similarly, $x_1 = F_{2\to1}^*(x_2) = G_1^*(E_2^*(x_2))$. Thus the image-to-image translation problem in UNIT is that of learning $F_{1\to2}^*$ and $F_{2\to1}^*$. For optimal learning of $F_{1\to2}^*$ and $F_{2\to1}^*$, it is necessary that there exist a cycle-consistency constraint ([13], [31]): $x_1 = F_{2\to1}^*(F_{1\to2}^*(x_1))$ and $x_2 = F_{1\to2}^*(F_{2\to1}^*(x_2))$. The shared latent space assumption implicitly enforces this constraint. The next section describes the idea in further details.

3.2.1 Framework

UNIT framework leverages recent progress in variational autoencoders (VAEs) ([15], [17], [24]) and generative adverserial networks (GANs) ([8], [19]). The framework is composed of 6 subnetworks: two domain image encoders E_1 and E_2, two domain image generators G_1 and G_2, and two domain adversarial discriminators D_1 and D_2. The framework learns the bidirectional translation in a single shot. The following paragraphs expound upon the roles of each of the networks.

VAE: The VAE for domain X_1, denoted by VAE_1, consists of the encoder-generator pair (E_1, G_1). For an input image $x_1 \in X_1$, the VAE_1 first maps x_1 to a code in a latent space Z via the encoder E_1. This code is randomly perturbed and subsequently decoded via the generator G_1 to reconstruct the input image. It is assumed that the components of Z are conditionally independent and sampled from a Gaussian distribution with unit variance. Thus the output of the encoder is a mean vector $E_{\mu,1}(x_1)$ and the distribution of the latent code is z_1, given by $q_1(z_1|x_1) \equiv \mathcal{N}(z_1|E_{\mu,1}(x_1), I)$, where I is an identity matrix. Then, the reconstructed image is $\tilde{x}_1^{1\to1} = G_1(z_1 \sim q_1(z_1|x_1))$. Similarly for domain X_2, the VAE denoted by VAE_2, constitutes of pair (E_2, G_2). Here the output of the encoder E_2 is a mean vector $E_{\mu,2}(x_2)$ and the distribution of the latent code z_2 is given by $q_2(z_2|x_2)\mathcal{N}(z_2|E_{\mu,2}(x_2), I)$. The reconstructed image in this case is $\tilde{x}_2^{2\to2} = G_2(z_2 \sim q_2(z_2|x_2))$.

This formulation of VAE is difficult to train via standard backpropagation as the sampling operation is non-differentiable. This can be solved by utilizing the reparameterization trick [15]. It reparameterizes the sampling operation as a differentiable operation using auxiliary random variables. This allows the VAEs to be trained using standard backpropagation. In the formulation discussed above, the sampling operations $z_1 \sim q_1(z_1|x_1)$ and $z_2 \sim q_2(z_2|x_2)$ can be implemented via $z_1 = E_{\mu,1}(x_1) + \eta$ and $z_2 = E_{\mu,2}(x_2) + \eta$, respectively. Here, η is a random vector with a multivariate Gaussian distribution: $\eta \sim \mathcal{N}(\eta|0, I)$.

Weight sharing. To make the VAEs converge via the shared latent space assumption, a weight sharing constraint is enforced. Since the last few layers of E_1 and E_2 are responsible for extracting high level semantic representations, the weights of these layers are shared between the encoders. Similarly, the first few layers of G_1 and G_2 are responsible for decoding the high level representations for the purpose of reconstructing the input images. Thus, the weights of those layers are shared between the generators.

It is worth noting the weight sharing constraint is not sufficient to guarantee that a pair of corresponding images from two domains will have the same latent code. Since we are working in an unsupervised setting, there cannot exist a pair of corresponding images from two domains such that they can train the network to output the same latent code. In general, the latent codes for a pair

of corresponding images will be different. Even in the rare case where the latent code is same, the components of the code can have different semantic meanings in different domains. This could lead to the generators to reconstruct two different images despite the same latent code. Although not possible through traditional methods, recently with progress in adversarial training techniques, as described in later sections, it is possible to map a pair of corresponding images from two domains to a common latent code using E_1 and E_2 respectively, and inversely, map a latent code to a pair of corresponding images in the two domains using G_1 and G_2 respectively.

The shared latent space assumption enables the translation of an image x_1 in X_1 to an image in X_2 through an information processing stream that is modeled by $G_2(z_1 \sim q_1(z_1|x_1))$. Such an information processing stream is termed as image translation stream and there exist two such streams: $X_1 \rightarrow X_2$ and $X_2 \rightarrow X_1$. The framework performs joint training of these streams with the image reconstruction streams from the VAEs. A pair of corresponding images, $(x_1, G_2(z_1 \sim q_1(z_1|x_1)))$ forms, once such a pair can be encoded to the same latent code and a latent code can be decoded to such a corresponding pair. Thus, for the unsupervised image-to-image translation problem discussed previously, $F^*_{1\rightarrow 2}$ is approximated by the composition of the functions E_1 and G_2 and $F^*_{2\rightarrow 1}$ is an approximation of the functions E_2 and G_1.

GANs. The framework consists of two generative adversarial networks: $\text{GAN}_1 = \{D_1, G_1\}$ and $\text{GAN}_2 = \{D_2, G_2\}$. For GAN_1, the discriminator D_1 learns to output true for the original images sampled from the first domain, and false for images reconstructed by G_1. The generator G_1 generates images from two types of streams: 1) $\tilde{x}_1^{1\to1} = G_1(z_1 \sim q_1(z_1|x_1))$ from the reconstruction stream and 2) $\tilde{x}_2^{2\to1} = G_1(z_2 \sim q_2(z_2|x_2))$ from the translation stream. The adversarial training is only applied to images from the translation stream $\tilde{x}_2^{2\to1}$, since the reconstruction stream can be trained with supervision. Similar processing is done for GAN_2 where D_2 outputs true for the original images sampled from the second domain and false for the images reconstructed by G_2. To further regularize the translation problem, the framework enforces the cycle-consistency constraint. This is easier to do as the shared latent space assumption implies the constraint.

Learning. Given the above, the framework jointly solves the learning of VAE_1, VAE_2, GAN_1 and GAN_2 for the three streams: the image reconstruction stream, the image translation stream and the cycle-reconstruction stream:

$$\min_{E_1,E_2,G_1,G_2} \max_{D_1,D_2} = \mathcal{L}_{\text{VAE}_1}(E_1, G_1) + \mathcal{L}_{\text{GAN}_1}(E_2, G_1, D_1) + \mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2)$$
$$\mathcal{L}_{\text{VAE}_2}(E_2, G_2) + \mathcal{L}_{\text{GAN}_2}(E_1, G_2, D_2) + \mathcal{L}_{\text{CC}_2}(E_2, G_2, E_1, G_1)$$

Here, the VAE objects are as given below and the VAE training aims for minimizing a variational upper bound

$$\mathcal{L}_{\text{VAE}_1}(E_1, G_1) = \lambda_1 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)}[\log p_{G_1}(x_1|z_1)]$$

$$\mathcal{L}_{\text{VAE}_2}(E_2, G_2) = \lambda_1 \text{KL}(q_2(z_2|x_2)||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_2 \sim q_2(z_2|x_2)}[\log p_{G_2}(x_2|z_2)]$$

where the hyperparameters λ_1 and λ_2 balance the importance of the objective terms and KL divergence is needed to penalize any deviation of the distribution of the latent code from the prior distribution. As discussed in [15], the regularization makes it easy to sample from the latent space. Here, the Laplacian distribution is used to model p_{G_1} and p_{G_2}. Since we're minimizing the negative log-likelihood term, this has the same effect as minimizing absolute distance between the original and it's reconstructed image. The prior distribution is a zero mean Gaussian: $p_\eta(z) = \mathcal{N}(z|0, I)$.

The conditional GAN objective functions are given by

$$\mathcal{L}_{\text{GAN}_1}(E_2, G_1, D_1) = \lambda_0 \mathbb{E}_{x_1 \sim P_{X_1}}[\log D_1(x_1)] + \lambda_0 \mathbb{E}_{z_2 \sim q_2(z_2|x_2)}[\log(1 - D_1(G_1(z_2)))]$$

$$\mathcal{L}_{\text{GAN}_2}(E_1, G_2, D_2) = \lambda_0 \mathbb{E}_{x_2 \sim P_{X_2}}[\log D_2(x_2)] + \lambda_0 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)}[\log(1 - D_2(G_2(z_1)))]$$

These ensure that the translated images are as close to the target domain images as possible. λ_0 weighs the individual of these objective functions.

To model the cycle-consistency constraint, a VAE-like objective function is utilized:

$$\mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2) = \lambda_3 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) + \lambda_3 \text{KL}(q_2(z_2|x_1^{1\to 2})||p_\eta(z)) -$$

$$\lambda_4 \mathbb{E}_{z_2 \sim q_2(z_2|x_1^{1\to 2})}[\log p_{G_1}(x_1|z_2)]$$

$$\mathcal{L}_{\text{CC}_2}(E_2, G_2, E_1, G_1) = \lambda_3 \text{KL}(q_2(z_2|x_2)||p_\eta(z)) + \lambda_3 \text{KL}(q_1(z_1|x_2^{2\to 1})||p_\eta(z)) -$$

$$\lambda_4 \mathbb{E}_{z_1 \sim q_1(z_1|x_2^{2\to 1})}[\log p_{G_2}(x_2|z_1)]$$

where the negative log-likelihood objective terms ensure that a doubly translated image ends up resembling the input one. The KL divergence terms are used to apply a penalty to latent codes that deviate from the prior distribution in the cycle-reconstruction stream. λ_3 and λ_4 balance contributions of the two objective terms respectively.

Just as with training with GANs, learning in the framework relies on optimization to find a saddle point by solving a minimax problem, which can be considered as a two player zero-sum game. This can be visualized as two teams: first team consisting of the encoders and generators of the framework and second team consisting of the adversarial discriminators. Not only does the first team aim to defeat the second team, it also has the responsibility of minimizing the VAE as well as cycle-consistency losses.

The end result of this learning is two translation functions that are obtained by assembling a subset of sub-networks. These are $F_{1\to 2}(x_1) = G_2(z_1 \sim q_1(z_1|x_1))$

that translates images from domain X_1 to X_2 and $F_{2\to1}(x_2) = G_1(z_2 \sim q_2(z_2|x_2))$ that translates images from X_2 to X_1. This comprises the framework for Unsupervised Image-to-image Translation.

Chapter 4

Methods and Results

Although synthetic datasets do a good job in capturing the scene abstractions such as objects and their positions, there are differences, particularly in lighting and textures that make them look different from images that are captured in a real world setting. These differences introduce a domain gap that limits generalization. Even though the domain gap is primarily due to low level pixel differences and may not extend to higher level abstractions of a scene, they nevertheless reduce performance when transferring the learned model. If we want to save the expense of annotating large amounts of real world data, we need to bridge this domain gap to improve transferability. Similar to style transfer, wherein we transfer the "style" of one image onto another image to achieve new and unique variations of the target image, we can transfer the pixel level distribution of one dataset onto

another to create a new dataset that has the same content as the target, but with the "style" of the source dataset.

In our case, we leverage the UNIT framework described in Chapter 3 to transform the synthetic images to look more like real world images from the test set. Using the training script at `https://github.com/mingyuliutw/UNIT` provided by the authors of [18], we trained two UNIT models, one with GTA5 as source domain and other with SYNTHIA as source domain. In both cases, the target domain was the cityscapes dataset. Both the source and target domain images were downsized to 800×475 pixels and the models were trained for 100k iterations. The generators and discriminator were optimized using Adam optimizer. Figures 4.2 and 4.3 show examples of transformed images using this method along with the original synthetic images. As seen, the transformed images capture the look and feel of real world images (compare them to Cityscapes images in Figure 2.2). We then train the segmentation model on this transformed dataset and test it on the real world Cityscapes dataset.

In the next few sections we discuss results obtained using the above methodology.

4.1 Experimental Protocol

We use DeepLab-v3+ [4] as our segmentation model for all our experiments. The backend for the model is selected to be MobileNet [27]. For training, we use Adam optimizer [14], with an initial learning rate set to $1e-4$. The ground truth labels are one-hot encoded and to better utilize hardware resources during training, we downsize the images to 800×475 pixels and set the batch-size to 2. For inference, we test the model on the Cityscapes validation set, which needs the evaluation images to be 2048×1024 pixels. For this reason, we upsample the images obtained during inference to this size. We also evaluate the effect of the different loss functions on training. The two loss functions that we tested are cross-entropy [26] and Dice loss [21], [28].

While cross-entropy loss performs well in classification problems, it tends to fare poorly for segmentation since it encounters severe class imbalance. Cross-entropy loss for segmentation is calculated as log loss, summed over all possible classes:

$$L_{CE} = -\sum_{classes} y_{true} \cdot \log(y_{pred})$$

where y_{true} is the ground truth map and y_{pred} is the predicted segmentation map. This evaluates the class predictions for each pixel individually and then

averages over all pixels. Thus, pixels belonging to underrepresented classes don't always get a significant error signal that can be backpropagated during training. This causes the network to get trapped in a local minima and become biased towards the majority classes.

To alleviate this issue, the Dice coefficient loss has recently become popular to train models for segmentation problems. An important property of Dice loss is that it estimates the Intersection over Union (IoU) metric, which is more relevant for segmentation problems. Essentially, it tries to measure the overlap between ground truth and the predicted segmentation maps. Dice loss as used in this work is defined as:

$$L_{Dice} = 1 - \sum_{classes} \frac{2 \cdot y_{true} \cdot y_{pred}}{y_{true}^2 + y_{pred}^2}$$

For upper-bound baseline results, we train the segmentation model on Cityscapes training set and perform inference using this model. This is described in the next section. We then evaluate the performance of a model trained on synthetic images and a model trained on photorealistic versions of these synthetic images.

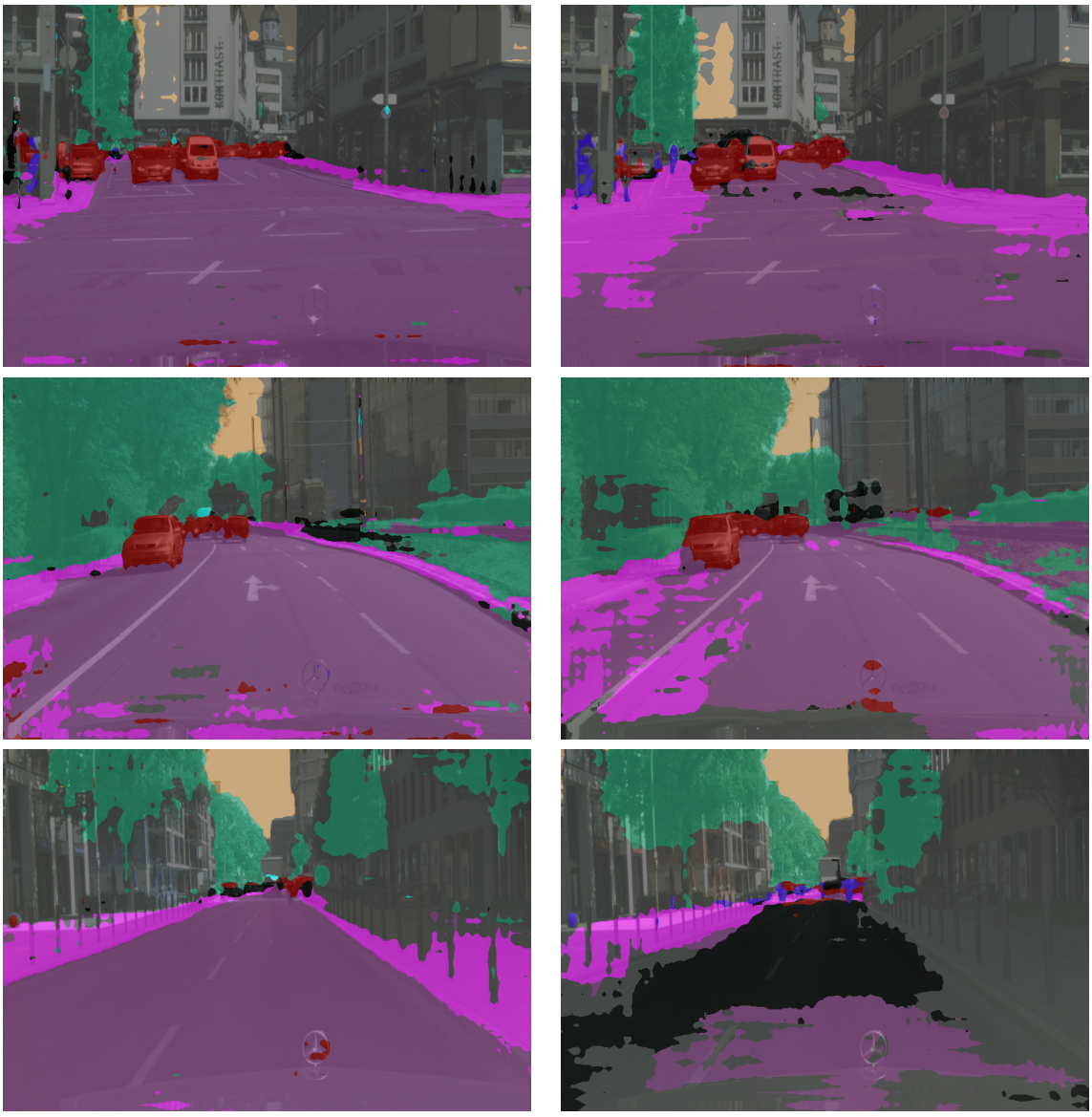
	road	sidewalk	building	vegetation	sky	person	car
Dice loss	0.963	0.733	0.865	0.866	0.871	0.554	0.869
Cross-entropy	0.960	0.703	0.847	0.859	0.911	0.589	0.843

Table 4.1: Baseline results of cityscapes dataset. The figures represent mIoU score across the Cityscapes test set

4.2 Baseline results

We first detail results obtained from training DeepLab-v3+ segmentation model on the Cityscapes training set. This serves as a reference to measure the performance of subsequent models. For easier comparison, Table A.1 in Appendix A combines results on all datasets and different models in a single table.

The model was trained on Cityscapes training set and evaluated on the validation set using the Cityscapes evaluation script. Table 4.1 details the performance of DeepLab-v3+ trained with two loss functions: Dice loss and cross-entropy loss. Training not only converged faster with Dice loss, but also the boundaries are sharper compared to cross-entropy loss, as can be seen in Figure 4.1.



(a) (b)

Figure 4.1: Results of training DeepLab-v3+ with Dice loss (column (a)) and cross-entropy loss (column (b))

4.3 Training with SYNTHIA images

	road	sidewalk	building	vegetation	sky	person	car
Dice Loss	0.666	0.037	0.675	0.650	0.658	0.278	0.264
Cross-Entropy	0.366	0.180	0.506	0.715	0.794	0.315	0.374

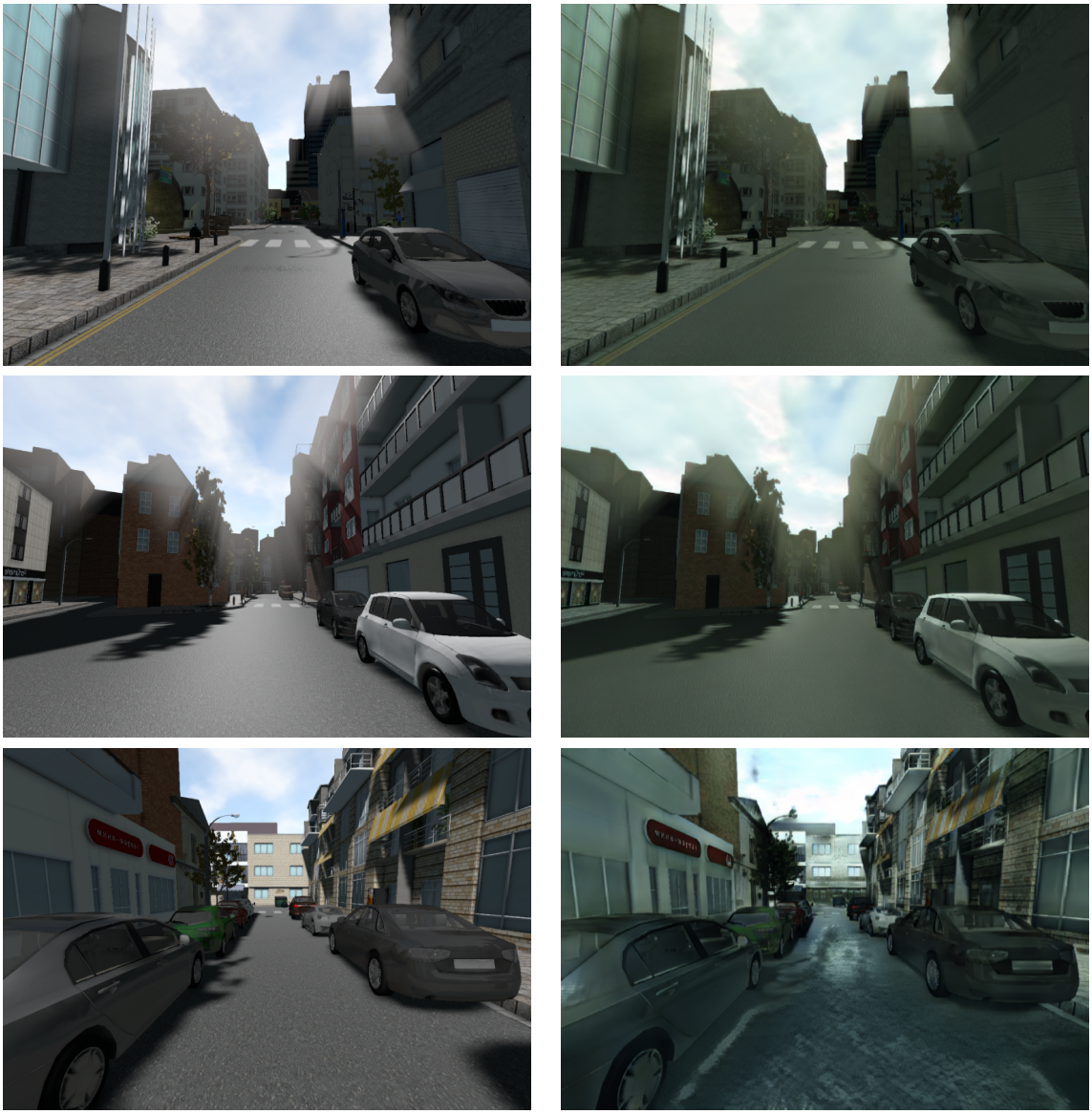
Table 4.2: Performance of DeepLab-v3+ model trained on original SYNTHIA dataset

	road	sidewalk	building	vegetation	sky	person	car
Dice loss	0.847	0.435	0.671	0.625	0.650	0.332	0.605
Cross-entropy	0.507	0.202	0.684	0.727	0.750	0.374	0.535

Table 4.3: Performance of DeepLab-v3+ model trained on photorealistic images from SYNTHIA

Next, we train a DeepLab-v3+ model on the original SYNTHIA images. The particular sequence that we used is SYNTHIA-RAND. The training setup is same as above. Table 4.2 describes the results for this setup.

We obtain more photorealistic data by transforming the source images, in this case the SYNTHIA-RAND dataset, using the UNIT framework. A sample of the source and transformed images is shown in Figure 4.2. As we can see, this transformation captures the look and feel of Cityscapes quite well. In theory, this should help close the domain gap to a certain extent. Results in Table 4.3 show that this is indeed the case.



(a) Original (d) Transformed

Figure 4.2: Sample of images showing how the photo-realistic transformation looks like. Column (a) are the original images from SYNTHIA-RAND dataset and column (b) are the photorealistic versions of them.

4.4 Training on GTA5 dataset

Since GTA5 leverages the high quality images derived from a commercial game, its performance tends to be better than the SYNTHIA original images. In other words, the domain gap between original GTA5 images and real world data is smaller to begin with. Table 4.4 shows the results of evaluating a model trained on the original GTA5 images. Transforming these images to be more photorealistic helps bring the performance very close to those of baseline results, as demonstrated in Table 4.5.

	road	sidewalk	building	vegetation	sky	person	car
Dice loss	0.266	0.203	0.658	0.652	0.496	0.058	0.167
Cross entropy	0.054	0.011	0.353	0.520	0.528	0.103	0.330

Table 4.4: Performance of DeepLab-v3+ model trained on GTA5 original dataset

	road	sidewalk	building	vegetation	sky	person	car
Dice loss	0.898	0.435	0.751	0.776	0.746	0.650	0.726
Cross entropy	0.852	0.228	0.714	0.762	0.762	0.260	0.619

Table 4.5: Performance of DeepLab-v3+ model on photoreal GTA5 dataset

Table 4.5 shows the performance of DeepLab-v3+ trained on the photoreal version of the GTA5 dataset. A sample of the transformed images are shown in Figure 4.3.

(a) Images (b) Transformed

Figure 4.3: Sample of images showing how the transformation looks like. Column (a) are the original images from the GTA5 dataset and column (b) are their photorealistic versions.

4.5 Discussion

	road	sidewalk	building	vegetation	sky	person	car
Cityscapes	32.64	5.39	20.21	14.10	3.56	1.08	6.19
SYNTHIA	18.68	19.04	29.32	10.41	6.80	4.31	3.92
GTA5	37.76	3.93	13.50	7.84	12.69	0.13	1.84

Table 4.6: Distribution of labels in each dataset (numbers are percent of pixels per class in entire dataset)

As seen in Sections 4.3 and 4.4, performance improves considerably when we train the segmentation model on images that are transformed to look more like Cityscapes images. When trained on original images, the categories "person" and "car" suffer a lot. This is because there aren't many instances of these classes for the model to generalize. As can be seen in Table 4.6, these categories are quite underrepresented, especially in GTA5. However, these still see a performance improvement when trained with photorealistic images.

Between Dice loss and cross-entropy loss, Dice loss clearly helps the model learn correct delineations amongst objects, even though the difference between these losses is not readily apparent during training as loss figures are not directly comparable. The performance improvement with Dice loss is due to indirect optimization of Intersection over Union (IoU) score of the classes, which enforces spatial constraints over the model during training.

Chapter 5

Conclusion

As deep learning models get more sophisticated and increase in capacity, the demand for training data will shoot up exponentially. For problems in areas like semantic and instance segmentation, this could pose as a bottleneck as annotating every object pixel-wise is labor-intensive and expensive. This calls for other ways to tackle such problems.

As we see here, games and simulators are practically an infinite source of such training data. The data is not only easy to obtain, but can model scenarios that would be impossible to replicate in real world. This provides a means to accelerate research and development in the field.

However, as demonstrated in this work, simply having this simulated data is not sufficient to guarantee a good performance in the final model. The problem of

domain gap stymies the generalization capabilities of many deep learning models when the test data that inference is performed on is different from the training data. For a problem in semantic segmentation, especially one that is performed on high-resolution input images, even minor differences in lighting and textures can cause significant drop in performance.

Until such time that segmentation methods are general enough to bridge this domain gap themselves, we'll need to rely on train or test time data augmentation, such as the one shown in above. As we see from results in the previous chapter, training data that looks as close as possible to test data can provide large improvement on segmentation accuracy without the need to change the model or its architecture. Although it may be prohibitive to have different models for different test distributions, for many practical applications this process needs to be performed only once. As such, it can greatly improve accuracy without needing a lot of effort from the practitioner's end.

This work can progress in many future directions. For starters, having a renderer that can produce realistic frames that look similar to test distribution will greatly improve the efficiency of the entire process and can do away with the need to manually transform training data. Additionally, such transformations can be built directly into the segmentation model to provide an end-to-end solution to this particular problem.

In conclusion, in this thesis, we show the effect of domain gap in segmentation problems by means of using a deep model trained on synthetic images to perform inference on real world data. We then show a way to bridge this domain gap by transforming the synthetic data to have a more photorealistic look using state-of-the-art techniques in generative modeling. The results thus obtained are promising enough to warrant further investigation in this direction and we look forward to seeing more such work being done to overcome the curse of domain gap and improving the generalization abilities of deep models for problems in semantic segmentation.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 40(04):834–848, apr 2018.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[5] 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.

[6] Epic Games. Unreal engine. `https://www.unrealengine.com/en-US/`.

[7] Rockstar Games. Grand theft auto v, 2014.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[9] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] Galen Hunt and Doug Brubacher. Detours: Binary interception of win32 functions. In *Third USENIX Windows NT Symposium*, page 8. USENIX, July 1999.

[12] Baldur Karlsson. Renderdoc.

[13] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1857–1865. JMLR.org, 2017.

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

[15] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[17] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1558–1566. JMLR.org, 2016.

[18] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017.

[19] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 469–477, USA, 2016. Curran Associates Inc.

[20] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[21] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[24] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1278–II–1286. JMLR.org, 2014.

[25] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[28] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In M. Jorge Cardoso, Tal Arbel, Gustavo Carneiro, Tanveer Syeda-Mahmood, João Manuel R.S. Tavares, Mehdi

Moradi, Andrew Bradley, Hayit Greenspan, João Paulo Papa, Anant Madabhushi, Jacinto C. Nascimento, Jaime S. Cardoso, Vasileios Belagiannis, and Zhi Lu, editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing.

[29] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge: Research articles. *J. Robot. Syst.*, 23(9):661–692, September 2006.

[30] Unity Technologies. Unity. `https://unity3d.com/`.

[31] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Oct 2017.

Appendix A

Results

		road	sidewalk	building	vegetation	sky	person	car
Baseline	Dice	0.963	0.733	0.865	0.866	0.871	0.554	0.869
	Cross entropy	0.960	0.703	0.847	0.859	0.911	0.589	0.843
GTA original	Dice	0.266	0.203	0.658	0.652	0.496	0.058	0.167
	Cross entropy	0.054	0.011	0.353	0.520	0.528	0.103	0.330
GTA photoreal	Dice	0.898	0.435	0.751	0.776	0.746	0.650	0.726
	Cross entropy	0.852	0.228	0.714	0.762	0.762	0.260	0.619
SYNTHIA original	Dice	0.666	0.037	0.675	0.650	0.658	0.278	0.264
	Cross entropy	0.366	0.180	0.506	0.715	0.794	0.315	0.374
SYTNHIA photoreal	Dice	0.847	0.435	0.671	0.625	0.650	0.332	0.605
	Cross entropy	0.507	0.202	0.684	0.727	0.750	0.374	0.535

Table A.1: Results of trained on all datasets for easier comparison. It can be seen that results on GTA5 with photoereal transformation approach the accuracy of baseline results