Domain Adaptation Semantic Segmentation for Self-Driving Cars

Yichen Jiang[1*]

[1]University of California, San Diego
[*]709851689@qq.com

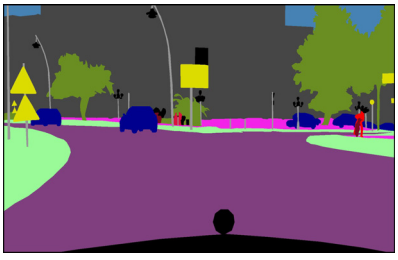
Abstract: As an extension of the traditional deep learning network, semantic segmentation aims to generate a class label for each pixel in the input image, thus grouping up together pixels that possess the same semantic label. It is always one of the key topics in computer vision domain, and it is also a vital concept in building self-driving automobile systems. One problem often met in real-life semantic segmentation problem is that researchers often can't provide a large enough dataset containing all diversely different scenes around the world, causing the model to underfit. In this article we show you how we use VGG16 deep learning model [7] to build a Fully Convolutional Network [5] as to achieve pixel-wise semantic prediction, and how we addressed aforementioned limitation by training the model using video game scenes from GTA5. Since there are style-level discrepancies between the game scenes and the real life scenes, we utilize Adversarial Domain Adaptation [2] to overcome problems caused by such domain shifts and to make the model perform better when working in real life scenes.

Computing Methodologies • Machine learning • Machine learning approaches• Neural networks
Additional Keywords and Phrases: Computer Vision, Adversarial Domain Adaptation, Semantic Segmentation, Fully Convolutional Network, Deep Learning, Self-Driving Car

1. Introduction

Being a fundamental keystone in the self-driving automobile system, semantic segmentation is also one of the most challenging tasks in the world of computer vision. The concept of semantic segmentation is to take the input of a real-world scene and to assign each pixel of the original image a certain semantic class label.

The purpose of semantic segmentation is to classify parts of the image together which belong to the same object, making it crucial to self-driving automobiles since accurate real-time objects detection and classification lay the foundation for scenario comprehension and subsequent decision of the automobiles (See Figure 1).



(a) Input Image

(b) Ground Truth Semantic Segmentation

Fig. 1. *Example of semantic segmentation*

In this project, we use fully convolutional network (FCN), with VGG16 [7] being the feature extractor model, to achieve semantic segmentation. FCN could take in input images of arbitrary sizes, and it uses "skip" layers to restore the fine and shallow location information while keeping coarse and deep feature information [5]. A large portion of the semantic segmentation problem still relies on the traditional class prediction network. VGG (Visual Geometry Group), for example, is a popular feature extracting model, famous for its utilization of smaller convolutional kernels and its potential to build a deeper and correspondingly stronger network [7].

Another limitation is that due to the diversely different traffic scenes around the world and the shortage of suitable real-world landscape images, training the model directly on the real-world dataset would cause the model to underfit. To address this problem, we transfer the "performance stage" to the video game domain. The game we choose is Grand Theft Auto V (GTA5), since it's famous for its extremely realistic in-game landscapes. Moving the training setting to game domains ensures that we can have as many suitable traffic scenes as we need to model the deep network. To enable the model trained on game domains to work successfully on real traffic scenes, we utilize domain adaptation (DA) which will be explained in detail in the following section.

The performance of the whole model is tested and shown in the fourth section.

2. Related Works

Being a relatively new research area which was first proposed in 2012 [1], semantic segmentation quickly gained attentions worldwide and flourished in 2015 with the advent of Fully Convolutional Network, which enabled the learning network to take in input images of arbitrary size and to restore the image

https://doi.org/10.1049/icp.2023.1892

size at the end of the whole model [5]. At the same time, as the daily development of the self-driving technology, scientists turned to semantic segmentation to help cars actually "see" and "understand" the scenes in front of them. However, one of the greatest problems met in this domain was that as the deep learning model became deeper and deeper in recent years, there weren't sufficient real-life datasets to train the oceans of parameters in the model well enough. That was how we turned our eyes to traffic scenes in realistic video games, which could generate a dataset as large as we want, and also to the domain adaptation technology that could fill the gap between the traffic scenes seen in real-life and in video game.

Our work drew on the previous success of fully convolutional networks for pixel-wise class prediction. We also utilized the concept of domain shift. Domain shift referred to the concept when the training data and real data were coming from different domains. Domain adaptation (DA) was used to transfer the deep model to work on the target domain from the source domain.

Fully convolutional networks Fully convolutional network (FCN) is one of the most widely used network for semantic segmentation since its born. It is built on traditional class classification network, replacing the last fully-connected layers with their equivalent convolutional layers.

FCN could take in input images of arbitrary sizes, which is an advantage that traditional network network (like LeNet [8], AlexNet [1]) does not possess. By introducing the concept of upsampling, with unconvolution being the most often used upsampling technique in practice, FCN is able to restore the size of the output from convolutional layers to match the original input size. Such upsampling brings back the detailed features from shallow structure, which is a key step in fine pixel-wise semantic segmentation. Adding upsampled outputs from shallow layers to those from deep layers makes sure that the network extract deep, coarse features but at the same time not lose shallow, fine spatial details of the input images [5].

Domain adaptation Many different domain adaptation (DA) strategies and models have been proposed ever since the domain shift problem was brought to the stage. Domain shift refers to the problem in which the model is trained on a certain source domain, and the same model needs to be transferred to work on inputs that come from another domain, the target domain.

Earlier works focused on matching the feature distributions between the two domains to achieve unsupervised domain adaptation. Typical methods of domain adaptation include metric learning, which chooses a certain metric to measure the (dis)similarity between the source domain and the target domain. Some other methods attempt to calculate an explicit mapping equation between the two domains [3]. We use adversarial learning as our domain adaptation method. Adversarial domain adaptation has three key structures, a deep feature extractor, a label predictor, and a domain classifier. The core idea of such adversarial DA is to make domain classifier combat against the feature extractor in order to extract the features that are most domain-invariant [2]. We will talk more about this idea in the next section.

3. Domain Adaptation Semantic Segmentation

3.1. The Datasets

We use two datasets in our project, with Cityscapes dataset being the target dataset and GTA5 dataset being the source dataset.

Cityscapes is a new large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities worldwide. Our project uses 5,000 high quality pixel-level annotated frames, with 3,475 training samples and 1,525 testing samples, as the target domain.

There are 8 groups of subjects and 30 diverse labels in total in Cityscapes dataset (Figure 2), making it a suitable target dataset (since we want to eventually make our model work in real traffic scenes) for adversarial learning and subsequent testings.

Fig. 2. *Cityscapes fine annotated frames*

The source dataset we choose is the in-game traffic scenes from Grand Theft Auto V, which is an epoch-making video game released in 2013. The city and landscapes in the game are built based on those in Los Angeles, so the details of the city scenes are marvellously realistic when compared to the real world. The game restores the real landscapes in Los Angeles, and it also has details as small as cracks and leaves on the street. The Grand Theft Auto V dataset we use is composed of 25,000 high quality in-game screenshots, including traffic scenes in cities, in rural areas, and on highways (Figure 3).

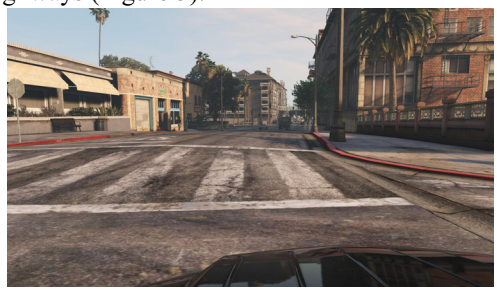


Fig. 3. *Grand Theft Auto V in-game traffic scenes*

3.2. Adversarial learning for domain adaptation

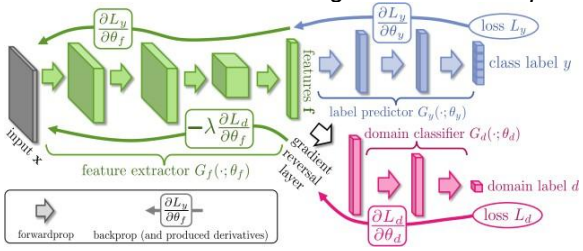


Fig. 4. *The architecture of adversarial domain adaptation model (figure from [2])*

We detail the model in our project here (See Figure 4). The problem we face in domain shift is that we have some inputs from both the source domain and target domain, and after training the model on samples partly from the source domain, the model needs to correctly predict label for samples from the target domain. Researchers use $S(x,y)$ and $T(x,y)$ to represent the source distribution and the target distribution, in which x represents the input into the model, and y represents the corresponding label of x. Usually, the two domains are complex and different, but they are also similar in certain aspects, thus we say that $T(x,y)$ is shifted from $S(x,y)$ by some domain shifts. The ultimate goal is to make the deep model predict the correct y given an input x from the target domain [2].

The adversarial discriminative domain adaptation method mixes the source and target datasets together, meaning the model would face oceans of training samples which come from both source domain and target domain. We use a domain label (a binary variable that is either 0 or 1) to mark the samples so the model knows which domain each sample is from [2].

The whole model is composed of three parts. The first part is a feature extractor, responsible for mapping the input image into a feature vector. We use VGG-16 [7] as our feature extractor in this project. The feature map is then fed into the label predictor to predict class labels. In adversarial learning model, there is also a domain classifier connected to the feature extractor and the domain classifier will take in the feature map and try to predict which domain the feature comes from. This classifier is also the key element in the model's 'combat'.

In the beginning, we feed inputs all the way throughout the "feed forward" structure (feature extractor plus la- bel predictor), and we optimize the parameters in these two stages in order to minimize the label prediction loss. By doing so, the system would have the basic function as other normal deep learning model that it could perform well in estimating the label for a given input.

However, we want to extract those features that have similar distributions in the two domains. Such features are called domain-invariant features. These features would make the model class prediction accuracy the same on two domains[6]. Instead of directly looking for (dis)similarities of a certain feature between two domains, which is complex and difficult, adversarial DA model uses the domain label prediction loss to measure the similarity of a feature in both domains. On the one hand, the domain classifier would optimize its parameters to minimize the domain prediction loss; On the other hand, the feature extractor updates its parameters to maximize the domain prediction loss. Maximizing the domain prediction loss enables the feature extractor to extract features that are most similar in two domains, while minimizing the domain prediction loss urges the domain classifier to develop sharper sense to distinguish features from different domains. Aforementioned ideas construct this 'adversarial' model, since the feature extractor is combating against the domain classifier through the training process, with one trying to 'fool' the classifier and the other studying to see through the tricks of the feature extractor. Thus, the whole system makes sure the feature extractor would find out features that are most domain-invariant [2].

3.3. FCN for semantic segmentation

To achieve semantic segmentation, we utilize the concept of fully convolutional network in the label predictor part. Fully connected layers used in traditional class prediction network can be seen as convolutional layers with kernels that cover the entire input region. Based on this concept, FCN replaces the fully connected layers with their equivalent convolution layers. Such architecture enables the whole network to generate not just a single class prediction, but an output of the same size as the input image, with each pixel in the output matrix being classified into a class. Doing so also facilitates the network to take input of any size, which traditional network fails to achieve [5].

After convolution layers, the size of the images are shrunk, making the output more and more coarse. Restor- ing dense pixel-level predictions from these coarse outputs requires the procedure called 'upsampling.' There are mul- tiple ways to upsample, including interpolation, deconvolution, and uppooling [5]. We use deconvolution (also called backwards convolution) in our model, which simply reverses the forward and backward passes of convolutions.

The third feature of fully convolutional network is its skip architecture. As the input goes through multiple convo- lution layers, deep features are extracted at the sacrifice of the loss of details. The output after the final prediction layer usually gets disappointingly coarse and the performance of semantic segmentation at pixel level would be limited after upsampling. FCN addresses the problem by upsampling the outputs coming from the shallow layers, which possess fine spatial detail information, and then combinning it with the final prediction output [5]. Such architecture builds links between the deep coarse output and the shallow fine output, making semantic seg- mentation fine-tuned in pixel-wise prediction.

4. Results

The metric to evaluate the performance of image seg- mentation and detection is Intersection over Union (IoU). IoU is used to evaluate the accuracy of an object detector or semantic segmentation model on a dataset. To calculate IoU, one needs the ground-truth semantic segmentation labels of a certain object and the predicted segmentation of the same object. Then we calculate IoU to be:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (1)$$

The numerator is the overlapped part between the ground-truth segmentation and predicted segmentation of the object, and the denominator is the overall area encompassed by both segmentation areas.

Following are the results after running the model (we use early stopping at 15000 data from GTA5 dataset to prevent over-fitting). In the training, we first test a prototype model without any domain adaptation which trains purely on the two datasets, as the control group. Then we train our model again using adversarial domain adaptation based on the two datasets, as mentioned before (Results shown in Table 1).

Table 1 Our model showed comparatively significant improve- ment in detecting objects like sidewalk, fence, building, rider, car, and motorcycle, and also slight improvement in detecting objects like pole and light. There are also drop-downs of IoU when using our model for detecting objects like vegetation, truck, or bus.

Object	Model without DA	Model with DA
road	57.65%	59.66%
sidewalk	14.82%	19.28%
building	55.22%	63.59%
wall	16.05%	14.29%
fence	9.67%	16.68%
pole	17.63%	18.65%
light	15.48%	16.16%
sign	3.34%	2.96%
vegetation	74.01%	68.51%
terrain	13.31%	8.43%
sky	60.02%	46.76%
person	39.45%	39.22%
rider	0.92%	1.78%
car	51.34%	57.0%
truck	15.71%	5.55%
bus	9.4%	0.38%
train	0.0%	1.64%
motorcycle	0.66%	5.91%
bicycle	0.0%	0.07%
mIoU	23.9%	23.5%

Utilization of adversarial learning improves the performance of detecting objects like cars, riders, buildings, and motorcycles, which are among the most important objects that needed to be detected in traffic scenes. However, IoU drops in terms of objects like vegetation, trucks, or buses, which happened probably because the in-game models of these objects are more crucially different from their counterparts in real life. The performances of both models in detecting person are similar.

Fig. 5. *Domain adaptation semantic segmentation performed on Cityscapes dataset. The first row represents the raw images. The second raw represents the ground-truth semantic segmentation (provided by Cityscapes dataset). The third row represents semantic segmentation achieved by our model.*

When visualizing the semantic segmentation results generated by our model, we can see that the model could correctly distinguish walking people and cars from other objects most of the time. In terms of the failure cases, the model seems to mistakenly treat terrains as vegetation sometime, and vice versa (Part of visual results are shown in Figure 5)

5. Conclusion

Importing domain adaptation concept to improve semantic segmentation facilitates the model to be able to use datasets that are large enough but not suitable in the first place. The model could train on these source datasets, and at the same time looking for features that are similar between the domains by using

adversarial learning. Such a model could come into use when the target dataset is not large enough to build a deep enough learning network.

Although the eventual accuracy of the model was somewhat lower than our expectation, we were still excited to witness the success of utilizing the concept of domain adaptation to solve a real-life problem. Such success meant that scientists could break the limit set by the size of the original target dataset, instead forcing the model to learn similar features from any other dataset.

We also felt impressed to see how our model could detect walking humans, other vehicles, and vegetations most of the time, with a relatively high accuracy since these objects were often the most important targets that we wanted a self-driving car to distinguish from the background when put into use.

5.1. Future work

The model is still not good enough, and we seek to build a better and a more realistic source dataset to train our model in future. The 'backbone' of our model (the feature extractor) also can be improved by utilizing better extractor model like Resnet [4]. We believe there is much room for promotion in future work.

6. References

[1] I. Sutskever A. Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks, 2012. 2

[2] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015. 2, 3, 4

[3] Li Ruonan Gopalan, Raghuraman and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach, 2011. 2

[4] S. Ren K. He, X. Zhang and J. Sun. Deep residual learning for image recognition, 2015. 5

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation, 2015. 2, 4

[6] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function, 2000. 3

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1

[8] J. Denker D. Henderson R. E. Howard W.Hubbard Y. LeCun, B. Boser and L. D. Jackel. Backpropagation applied to handwritten zip code recognition, 1989.