# Semantic Segmentation Domain Adaptation with Generative Model

Zhanghao Sun, Qiwen Wang, and Mi Jeremy Yu

## Introduction

In this project, we explore the domain adaptation problem in semantic segmentation. Semantic segmentation aims to assign each pixel a semantic label. Recently, CNN based models have achieved significant progress on this task. However, there's a large domain gap between synthetic and real-world data. This impairs the practical application of models trained on large and cheap synthetic datasets.

We set up adversarial models for this task. Our contributions are in three folds:

1. Implemented several multi-layer fusion discriminator architectures.
2. Proposed and implemented a category-level discriminator. Explored its potential advantage in category clustering.
3. Implemented an ensemble model to achieve state-of-the-art performance on GTA5-Cityscape dataset adaptation.).

## Problem Statement

Let $\{x_i\}_{i=1}^N$ be an input space of images and $\{y_i\}_{i=1}^N$ be the corresponding labels. We have two domains $\mathcal{D}_s$ and $\mathcal{D}_t$ following different distributions. In the scenario of unsupervised domain adaptation, we have access to the source domain labels $y_i^s$ but not the target domain labels $y_i^t$. Our objective is to learn a feature space $f$ that can close the gap between source and target domain. Specifically, our model is based on "AdaptSegNet"[2]. Instead of directly align the two domains in feature space, it uses a discriminator at the output stage, which back-props to align the feature space.

## Experimental Details

We trained our model on the synthetic GTA5 dataset consists of 24966 images with resolution $1914 \times 1052$ synthesized from the video game based on Los Angeles's city. We evaluate the model on real-world images from the Cityscapes dataset captured in cities around

## Method

We adopt Deeplab framework with ResNet 101 as the base segmentation network of the following methods.

- No Adaptation (Baseline): Train the segmentation generator $\mathbf{G}$ with source images.
- Single-level: Train a fully convolutional discriminator $\mathbf{D}$ using a cross-entropy loss for the source and the target segmentation generated from $\mathbf{G}$.
- Multi-level[2]: To adapt domain on feature level, we adopt a feature discriminator $\mathbf{D}_1$ and a classifier discriminator $\mathbf{D}_2$ on the classification prediction $f_s, f_t$ from the 2nd to the last feature representation and the output classification prediction $g_s, g_t$.
- Multi-layer fusion: Based on the multi-level model, we fuse the classification prediction $f_s, f_t$ from the 2nd to the last feature representation and the output classification prediction $g_s, g_t$ by a fusion function $T$ (i.e. concatenating or randomized inner product) to train the discriminator $\mathbf{D}$.

$$\mathcal{L}_{adv} = \lambda \left( \mathbb{E}_{x_i^s \sim \mathcal{D}_s} \log[\mathbf{D}(T(f_s, g_s))] + \mathbb{E}_{x_j^t \sim \mathcal{D}_t} \log[\mathbf{D}(T(f_t, g_t))] \right),$$

$$\text{where } T(f,g) = \begin{cases} f \| g & \text{(Concatenation)} \\ \frac{1}{\sqrt{d}} (\mathbf{R}_f f) \odot (\mathbf{R}_g g) & \text{(Randomized inner product)} \end{cases}.$$

- Category-level: The discriminator $\mathbf{D}$ uses a grouped convolutional layer to each channel of the prediction output $g_s, g_t$ to guarantee no interference between different categories in the discrimination process.

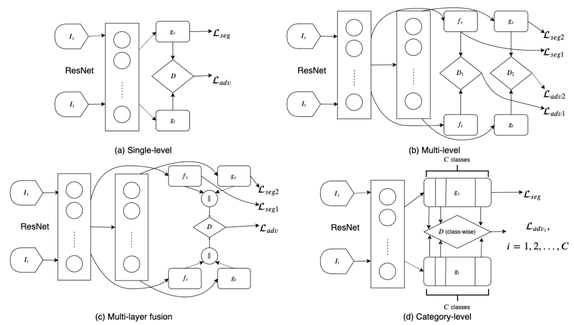



Figure 1: Network Architecture

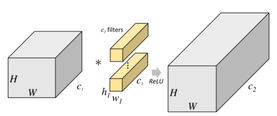


Figure 2: Convolutional layer in the discriminator of the single-level, multi-level and multi-layer fusion models.

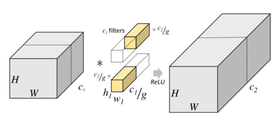
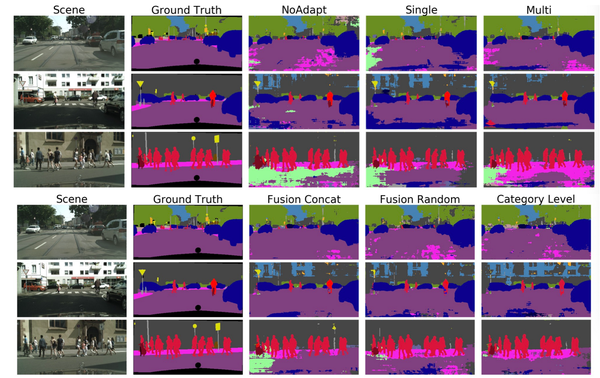


Figure 3: Group convolutions in the discriminator of the category-level model.

## Results and Discussion




| Method | road | sidewalk | building | wall | fence | pole | light | sign | veg | terrain | sky | person | rider | car | truck | bus | train | mbike | bike | mIoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No Adapt | 78.4 | 28.4 | 74.7 | 20.4 | 20.0 | 21.8 | 29.2 | 14.6 | 77.1 | 17.5 | 72.2 | 55.9 | 24.2 | 63.4 | 20.6 | 22.8 | 4.7 | 24.5 | **39.6** | 37.4 |
| Single-Level | **86.5** | 25.9 | 79.8 | 22.1 | 20.0 | 23.6 | 33.1 | **21.8** | 81.8 | 25.9 | **75.9** | 57.3 | 26.2 | 76.3 | 29.8 | 32.1 | **7.2** | 29.5 | 32.5 | 41.4 |
| Multi-Level | **86.5** | **36.0** | 79.9 | 23.4 | **23.3** | 23.9 | **35.2** | 14.8 | **83.4** | **33.3** | 75.6 | 58.5 | **27.6** | 73.7 | 32.5 | **35.4** | 3.9 | **30.1** | 28.1 | **42.4** |
| Fusion Cancat | 85.5 | 28.1 | **80.4** | 27.1 | 23.0 | 26.0 | 32.3 | 21.2 | 82.9 | 26.1 | 74.4 | **59.0** | 27.3 | **76.9** | **34.3** | 28.7 | 0.3 | 29.8 | 32.4 | 41.9 |
| Fusion Random | 84.4 | 34.7 | 77.7 | 25.6 | 20.4 | 24.8 | 27.8 | 17.3 | 81.4 | 30.1 | 75.7 | 57.6 | 20.8 | 71.2 | 27.2 | 29.7 | 0.5 | 28.1 | 30.1 | 40.3 |
| Categorical | 79.0 | 30.4 | 77.5 | **27.7** | 20.3 | **26.8** | 29.9 | 18.5 | 80.7 | 23.2 | 71.6 | 56.2 | 19.9 | 59.0 | 25.1 | 25.4 | 5.1 | 22.4 | 22.3 | 37.9 |

1. Category-level model results in more confident boundaries, as expected. However, due to lack of information from other classes, it can make severe mistakes.
2. Fusion model with simple concatenation is better than randomized fusion model. This is due to the limited dimension of $R_g$ and $R_f$. Segmentation problem has much larger feature space than that in classification problem [1]

## Future Works

1. Other implementations for category-level domain adaptation, Especially, models based on col-training.[3]
2. Multi-target domain adaptation can be formulated as a multi-objective optimization problem.

## References

[1] Mingsheng Long et al. "Conditional adversarial domain adaptation". In: *Advances in Neural Information Processing Systems*. 2018, pp. 1640–1650.

[2] Yi-Hsuan Tsai et al. "Learning to adapt structured output space for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7472–7481.

[3] Yuchen Zhang et al. "Bridging Theory and Algorithm for Domain Adaptation". In: *arXiv preprint arXiv:1904.05801* (2019).