TwinLossGAN: Domain Adaptation Learning for Semantic Segmentation

by

Yuehua Song

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Most semantic segmentation methods based on Convolutional Neural Networks (CNNs) rely on supervised pixel-level labelling, but because pixel-level labelling is time-consuming and laborious, synthetic images are generated by software, and their label information is already embedded inside the data; therefore, labelling can be done automatically. This advantage makes synthetic datasets widely used in training deep learning models for real-world cases. Still, compared to supervised learning with real-world labelled images, the accuracy of the models trained using synthetic datasets is not high when applied to real-world data.

So, researchers have turned their interest to Unsupervised Domain Adaptation (UDA), which is mainly used to transfer knowledge learned from one domain to another. That is why we can use synthetic data to train the model. Then, the model can use what it learned to deal with real-world problems. UDA is an essential part of transfer learning. It aims to make two domain feature distributions as close as possible. In other words, UDA is mainly used to migrate the learned knowledge from one domain to another, so the knowledge and distribution learned from the source domain feature space can be migrated to the target space to improve the prediction accuracy of the target domain.

However, compared with the traditional supervised learning model, the accuracy of UDA is not high when the trained UDA is used for scene segmentation of real images. The reason for the low accuracy of UDA is that the domain gap between the source and target domains is too large. The image distribution information learned by the model from the source domain cannot be applied to the target domain, which limits the development of UDA.

Therefore we propose a new UDA model called TwinLossGAN, which will reduce the domain gap in two steps. The first step is to mix images from the source and target domains. The purpose is to allow the model to learn the features of images from both domains well. Mixing is performed by selecting a synthetic image on the source domain and then selecting a real-world image on the target domain. The two selected images are input to the segmenter to obtain semantic segmentation results separately. Then, the segmentation results are fed into the mixing module. The mixing model uses the ClassMix method to copy and paste some segmented objects from one image into another using segmented masks. Additionally, it generates inter-domain composite images and the corresponding pseudo-label. Then, in the second step, we modify a Generative Adversarial Network (GAN) to reduce the gap between domains further. The original GAN network has two main parts: generator and discriminator. In our proposed TwinLossGAN, the

generator performs semantic segmentation on the source domain images and the target domain images separately. Segmentations are trained in parallel. The source domain synthetic images are segmented, and the loss is computed using synthetic labels. At the same time, the generated inter-domain composite images are fed to the segmentation module. The module compares its semantic segmentation results with the pseudo-label and calculates the loss. These calculated twin losses are used as generator loss for the GAN cycle for iterations. The GAN discriminator examines whether the semantic segmentation results originate from the source or target domain.

The premise was that we retrieved data from GTA5 and Synthia as the source domain data and images from CityScapes as the target domain data. The result was that the accuracy indicated by the TwinLossGAN that we proposed was much higher than the base UDA models.

Acknowledgements

First of all, I would like to express my very great appreciation to my parents for their endless support and love in my life.

Next, I would like to express my deep gratitude to my research supervisor, Professor Wonsook Lee, for her patient guidance, enthusiastic encouragement and helpful suggestions throughout my graduate studies.

Finally, I would also like to express my gratitude to my friends for their friendship and encouragement. Thankful for all that we encounter in life, whether happy or sad, it's all a treasure.

Table of Contents

List of Tables

List of Figures

Abbreviations

BN Batch Normalization 33

CNN Convolutional Neural Network iii, 4, 18, 19, 22, 24, 28, 31

CRF Conditional Random Fields 22, 24, 28

DA Domain Adaptation 2–4, 7, 22, 29–35, 42, 44, 45, 48

FCN Fully Convolutional Network 25, 48

FPN Feature Pyramid Network 25

GAN Generative Adversarial Network iii, iv, 2, 3, 5, 6, 16, 21, 22, 28, 34, 44, 48–51, 53, 56, 59, 60, 63, 72, 73

Iou Intersection-over-union 65, 67

MMD Maximum Mean Discrepancy 30, 31

UDA Unsupervised Domain Adaptation iii, iv, 3, 36, 61, 63, 72

Chapter 1

Introduction

Deep learning currently provides the main development direction for machine learning and artificial intelligence research, bringing revolutionary advances to diverse fields, including computer vision. In computer vision, deep learning has been prominent in image classification, object detection and semantic segmentation [7, 27, 30, 31, 78, 79] in recent years. Semantic segmentation, which associates labels or categories for each pixel of an image, aims to cluster pixels into semantically meaningful classes. One thing to note is that semantic segmentation does not separate instances of the same class; it only cares about the class of each pixel. In other words, if the input image has two objects of the same class, semantic segmentation does not distinguish them as separate objects.

There are two training mechanisms in the training process for semantic segmentation: supervised learning and unsupervised learning. Supervised learning derives the prediction function from labelled training data. The labelled training data means that each training instance includes both the input and the desired output [62]. Under the supervised learning mechanism, the model can learn the distribution of the data well because the model is supervised by the ground truth at the pixel level of the image so that the final model can achieve a high accuracy under this mechanism. The most popular deep learning models on semantic segmentation [6] achieve good results using supervised learning, but they also have their non-negligible drawbacks. Supervised learning relies on a large number of pixel-level image annotations. These annotations are usually performed manually and are time-consuming and labour-intensive, which blocks the use of supervised learning.

Unlike supervised learning, unsupervised learning infers conclusions from unlabelled training data. Unsupervised learning typically utilizes clustering and association, which can be used in the exploratory data analysis phase to discover hidden patterns or group

data [62]. Since unsupervised learning does not require a lot of manual labelling, a large number of unsupervised learning models have been proposed for the semantic segmentation of images in recent years.

There is also an issue of unseen images in the test stage. The trained models are not well generalized to unseen images. The reason is that there are differences between trained data and test data. As an example of city image data, the appearance of objects and scenes is not identical in different cities. Even the same city can have huge differences due to different lighting conditions. If these two data are from different domains, the performance degenerates even more.

Transfer Learning is a system that has the ability to recognize and apply knowledge and skills learned in existing domains/tasks to novel domains/tasks [81]. It aims to improve target learners' performance on target domains by transferring the knowledge contained in different but related source domains. For example, knowledge gained while learning to recognize cars could be applied when trying to recognize trucks.

In transfer learning, our existing knowledge is called the source domain, and the new knowledge to be learned is called the target domain.

The particular type of transfer learning we are concerned with is called Domain Adaptation (DA). The domain adaptation approach is to use knowledge from an abundantly labelled source domain to train an effective model for the target domain with limited or no labels while transferring the domain shift problem [77]. As a branch of unsupervised learning, DA aims to map data with different distributions of source and target domain into a feature space so that they are as close as possible in that space. The objective function trained on the source domain of the feature space can then be migrated to the target domain to improve the accuracy of the target domain.

DA methods are divided into three main categories. (i) Feature adaptation maps the source domain and target domain samples to the same feature space. It allows the samples to be "aligned" in this feature space. It is also the most common method. (ii) Instance adaptation considers that there are always some samples in the source domain that are similar to the target domain samples. So we assign different weights to all samples in the source domain during the training. The more similar the samples are to the target domain, the larger the weight will be. (iii) Model adaptation based on model parameters finds new parameters to make the model work better on the target domain by parameter migration.

Mei Wang and Weihong Deng [63] classify the application of domain adaptation into three categories in their survey. (i) The first category is discrepancy-based adaptation, mostly used for image classification [70]. (ii) The second category is adversarial learning-based adaptation, which is based on GAN [16] neural networks, and the trained models

are mostly used for scene analysis. (iii) The third category is data reconstruction-based adaptation. Since GAN models are trained using adversarial learning, this approach can help DA models align the distribution of source domain images and target domain images in the feature space.

When the images and labels of the source domain are visible to the model, the target domain images are also visible to the model. Still, the target domain data do not have labels. We consider this UDA [77, 82].

UDA is a kind of transfer learning that is used to solve the distribution bias of source domain and target domain sets. Various domain adaptation methods aim to apply the classifier learned from the source domain to the target domain without labelling in the target domain by learning the domain invariant features of source and target domains.

1.1 Problems and Our Solutions

We have also found many problems when applying deep learning to computer vision. Computer vision tasks often require fine-grained annotations in supervised learning, such as pixel-level category annotations, ID information to distinguish different instances (cars or people), and video motion information. These fine-grained annotations are time-consuming and labour-intensive. This kind of problem limits the model's learning ability and semantic segmentation accuracy.

In recent years, the use of synthetic data has become widespread. Synthetic data refers to computer-generated data, usually using game engines such as Unreal, Blender and Unity. The acquisition of new controllable samples is based on a priori knowledge, specifically by using simulated scenes to obtain specific images, videos and annotations. In summary, synthetic data can be generated quickly and accurately. The advantages of this type of data are:

- The large amount of data generated.
- The high quality of the labels.
- The diversity of the data.

Diversity means that the environmental conditions are varied, such as weather, time of day (morning and evening), light, etc. We want to explore the potential offered by

the availability of pixel-level labels of synthetic images and by the use of these images for training our model.

As supervised learning with labelling information generally performs better than learning without labelling, we consider a case where the model can be trained using the already labelled synthetic image in the source domain. Then, the trained model can be used to predict the unlabelled images in the target domain. However, because the distribution of source domain and target domain images sometimes differs greatly, the accuracy of the model trained with source domain images is not high when used for target domain image prediction, so DA is required here. Without DA, the trained model has very low accuracy with real-world images. The reason is that the model does not have a strong ability to segment images that have not been seen before. DA allows the model to predict real images using the characteristics of the knowledge learned from the synthetic dataset.

After testing some existing DA networks [14, 56], we found that the accuracy of these networks is very low for detecting small and large objects. The maximum accuracy for detecting large objects in a dataset called GTA5, such as walls, is currently only 32.7%. Additionally, for detecting small objects, such as fences and poles, the maximum accuracy is currently only about 25%. After finding these accuracy problems, it motivates us to propose a new model to address the lack of accuracy of traditional models for segmenting large and small objects.

Unlike image classification, feature adaptation for semantic segmentation may be affected by the complexity of high-dimensional images and requires encoding of different visual cues. This motivates us to create a new model that needs to be able to learn the high-dimensional semantic information of images well, and in which the learning process is relatively stable.

1.2 Our Model Overview

The main task of our model is to segment real street scene images. Our end-to-end Convolutional Neural Network-based (CNN-based) DA model for semantic segmentation is called TwinLossGAN. The model is divided into two parts.

The first part is the image fusion of the source and target domains, and the overall learning ability of the model is reduced because the labels of the target domain are not visible to the model and the image distributions of the source domain and target domains are different. Then, by using the mixing data method, the image distribution between source and target domains can be narrowed by image fusion between different domains.

The second part is the GAN-based segmenter, which solves the overhead problem caused by manual labelling and lays the foundation for the subsequent segmenter learning. The generator segments the images and computes the losses for both the source domain and target domain. That is where the twin losses come from. Twin losses serve to help the generator learn the features of both domains in the feature space. Then, the generator sends the segmented results to the discriminator, which is used to determine which domain the images come from. The generator here is used to learn domain-invariant features and perform domain adaptation. The discriminator is for adversarial learning with a generator. As the accuracy of generator segmentation increases, it becomes difficult for the discriminator to distinguish the difference between source domain images and target domain images.

GAN networks in our new model can first help the model obtain twin losses from the source domain and target domain. In this way, the generator can be well-trained. Second, the discriminator can help the model to migrate the learned distribution of synthetic images to the real images, which makes the model good at semantic segmentation of the real images. During the training process, the generator and discriminator have different learning abilities, which makes the model training process unstable. Nevertheless, the addition of the mixing data method helps the generator to improve its learning ability. Therefore, the whole training process is stable. After tens of thousands of iterations, the model can use the image features learned in the source domain to perform accurate semantic segmentation of the target and the real image.

Our technological innovation is a novel idea about a twin-losses network. Our goal is to improve the performance of DA models based on adversarial learning, and our main contributions include:

- Reduce the gap between the source and target domains by mixing objects in an image from different domains.
- One more loss in the real data where the partial labels are borrowed from the synthetic object in mixed images.

1.2.1 Publication

Yuehua Song and Wonsook Lee. MixedGAN: Facilitating GAN for Domain Adaptation Learning based on Mixing Different Domain Images for Semantic Segmentation. In 2022 IEEE International Conference on Semantic Computing (ICSC2022). Location: Laguna Hill, CA, United States. Conference dates: Jan 26–28, 2022.

1.3 Thesis Outline

This Thesis Outline is organized as follows.

- Chapter 2 introduces the background of machine learning and delves into the development of deep learning and some classical deep learning networks for semantic segmentation. Then, we discuss the background of domain adaptation. The solutions and problems of some existing models for treating inter-domain differences are discussed. Then, the basic principles and methods of image fusion are also presented.
- Chapter 3 first reviews the structure of the GAN network model and its theory. Then, the structure of our proposed model is described in detail. A detailed analysis of all the parts is also presented. At the end of the chapter, we introduce the most important part of the neural network model: the loss function.
- Chapter 4 discusses the performance evaluation results.
- Chapter 5 gives the conclusion of this thesis and discusses a few possible future research directions.

Chapter 2

Background and Literature Review

In this chapter, we present studies from various disciplines that are relevant to this work. In Sections 2.1, 2.2 and 2.3, we introduce the basic knowledge of machine learning and the foundation of deep learning – neural networks. Furthermore, we discuss deep learning. Then, in Section 2.4, we describe the specific image recognition task, which is semantic segmentation using deep learning. The methods used for semantic segmentation and some classical model implementations are discussed. Next, we discuss DA in Section 2.5 in detail, comparing different DA methods. In Section 2.6, we also compare various methods of image fusion. In Section 2.7, we discuss DA-based semantic segmentation models.

2.1 Machine Learning

Machine learning is the study of computer algorithms that can improve automatically by learning and gaining experience [36]. It is a branch of artificial intelligence. The basic idea of machine learning is that systems can learn from data, recognize patterns and make decisions with minimal human intervention [62].

There are three types of machine learning. The first type is unsupervised learning. Unsupervised learning refers to the automatic search for patterns and previously undetected information. It mainly deals with unlabelled data. The second type is supervised learning. Supervised learning is a machine learning task that infers a function from labelled training data. In supervised learning, each training data example is a pair consisting of an input object and its desired output value [38]. The third type is reinforcement learning. Reinforcement learning is a type of learning that can support people in decision making

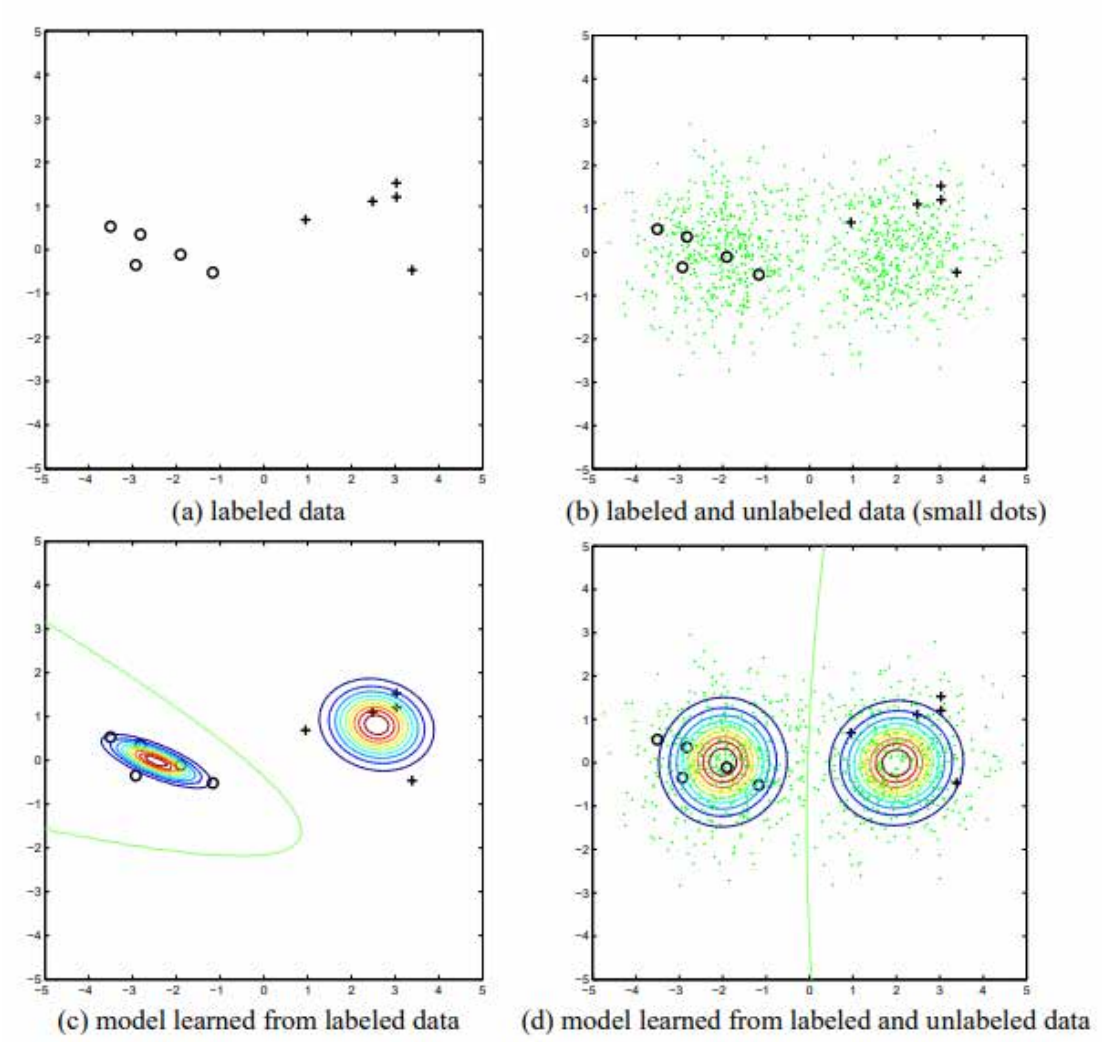

Figure 2.1: In binary classification problems, if each category has a Gaussian distribution, then we can use unlabelled data to help the model in parameter estimation. This image is from [80]. Permission is requested.

and planning. It is a feedback mechanism that rewards intelligent agents' actions and behaviours and facilitates learning through this feedback mechanism. In this context, an intelligent agent perceives the environment it is in, acts autonomously to achieve its goals, and may improve its performance or use knowledge through learning.

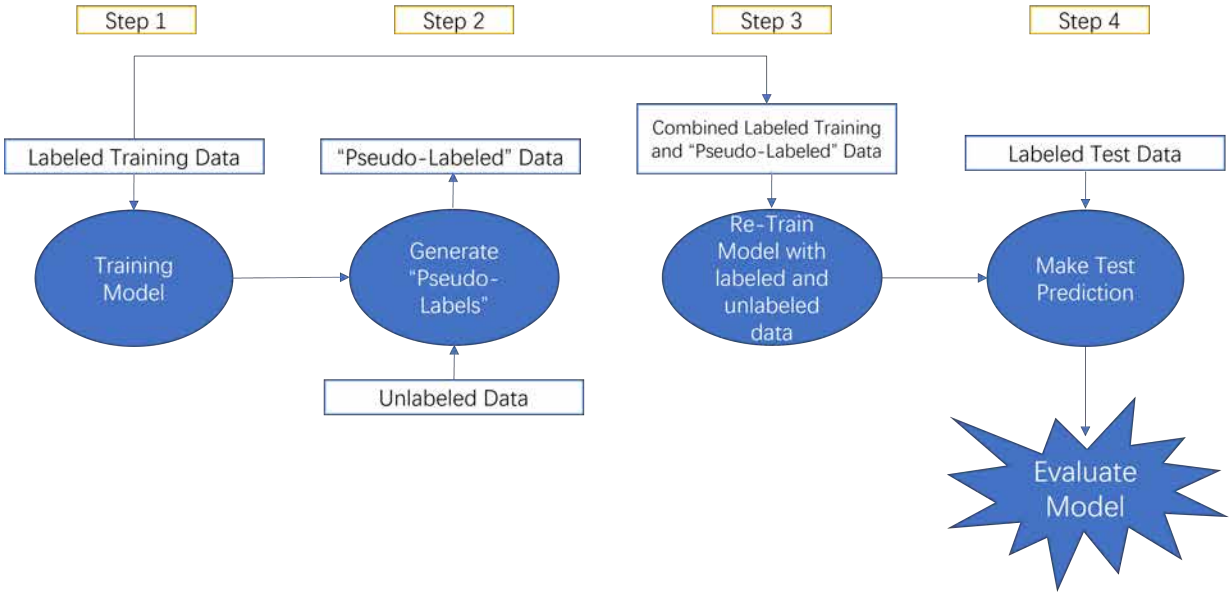

Figure 2.2: The self-training process. First, the labelled data are divided into training and test sets. The model is trained using the split training-set data. The trained model is used to predict the unlabelled data and generate pseudo-labels. The pseudo-labelled data are mixed with the training set, and the model is retrained. The model is evaluated using the test set after completing the training.

The semi-supervised learning algorithm is an algorithm that lies between supervised learning and unsupervised learning. It has been widely used in the field of machine learning. There are many categories of semi-supervised learning, and we will discuss some of them below [34].

1. Generative Models: Generating models is an old semi-supervised learning method [80]. It assumes a model $p(x, y) = p(y)p(x|y)$. where $p(x|y)$ is a mixture distribution, such as a Gaussian mixture model. When there is a large amount of unlabelled data, the mixture components can be identified. Then, ideally, we only need one labelled example for each component to fully identify the mixture distribution, see Figure 2.1.

2. Self-Training: Self-Training is a commonly used semi-supervised learning technique [80]. The training process begins by training a classifier using a small amount of labelled data. The classifier is then used to classify the unlabelled data. In general, the unlabelled data with the highest confidence and their predicted labels are added to the training set. The classifier is retrained. The whole process is repeated several times to achieve better training results. The self-training structure is shown in Figure 2.2.

2.2 Neural Network

As one machine learning algorithm, neural networks are the foundation of deep learning, the hottest research area in artificial intelligence nowadays. With its unique network structure and method of processing information, the neural network has led to brilliant achievements in many fields such as automatic control, combined optimization problems, pattern recognition, image processing and natural language processing. It is dedicated to learning patterns from data.

A neural network is an operational model. It consists of many nodes (or neurons) interconnected with each other. Each node represents a specific output function, called the activation function. Each connection between two nodes represents a weighted value of the signal, called weights. The neural network assigns weights to each element during training. Weight affects the influence a change in the input will have upon the output. The bias shifts the activation function to the left or the right. It helps the neural network to fit the input data better. The output of the network depends on the structure of the network, the connectivity of the network, the weight and the activation function. The design of a neural network is shown in Figure 2.3.

Backpropagation is a standard method for training neural networks. It is often combined with optimization methods (e.g., gradient descent). Backpropagation is done by computing the gradient of the loss function for the weights in the network. This gradient can be used to update the weights to minimize the loss function. The algorithm for backpropagation is shown in Algorithm 2.1.

The backpropagation process encounters two common problems: gradient vanishing and gradient explosion. The gradient vanishing problem arises because the gradient solution in the backpropagation process generates successive multiplications of derivatives and parameters. If the initial weights are small, the gradient near the input layer is almost zero. The neural network parameters are not updated. The gradient explosion problem

is due to the large initial weights, and multiple multiplications will lead to overflow. The model does not converge.

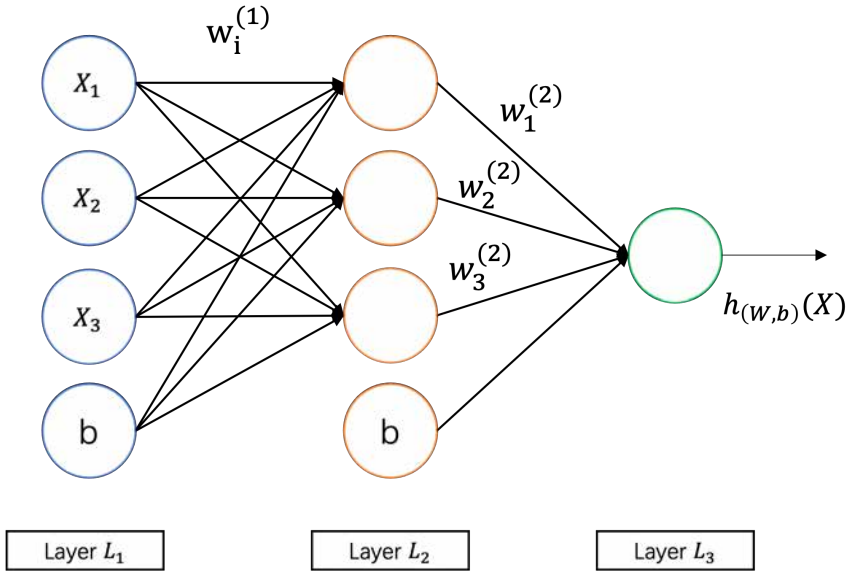

Figure 2.3: Neural network structure. The neural network shown in the figure has three layers, the input layer (L_1), hidden layer (L_2) and output layer (L_3). The input layer does not perform any operation, and it only represents the input of the neural network. The hidden layer is designed to extract the abstract features of the input data. The output layer is used to output the prediction results of the model. In the figure, w denotes weights. Weights can be thought of as the strength of the connection. Weight affects the influence a change in the input will have upon the output. In the figure, b indicates bias. Bias represents how far off the predictions are from their intended value.

2.2.1 Activation Functions

Among the many factors that influence the training of neural networks, the activation function plays an important role. There are five classical activation functions. They are the step function, the sigmoid function, the Tanh function, the ReLU function and the Leaky ReLU function.

Algorithm 2.1 Backpropagation algorithm (From Wikipedia)

Input: Initialize network weights (usually small random values)
  for Sample X in training set **do**
    $prediction = neural - net - output(network, X)$
    $Calculate\ the\ error\ of\ the\ output\ unit: (prediction - ground\ truth)$
    $Calculate \bigtriangledown w_h\ for\ all\ hidden\ layers\ to\ the\ weights\ of\ the\ output\ layer$
    $Calculate \bigtriangledown w_i\ for\ all\ input\ layers\ to\ the\ hidden\ layer\ weights$
    $Update\ network\ weights$
    $Stop\ when\ All\ samples\ are\ correctly\ classified\ or\ meet\ other\ stopping\ criteria$
  end for
  return *network*

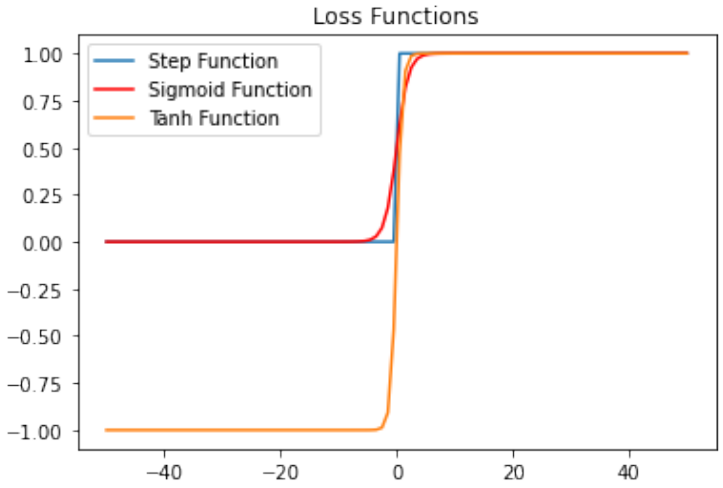

Figure 2.4: The blue line represents the step function, which has only two states, "activation" and "inhibition". Therefore, it is often used to create binary classifiers. The red line represents the sigmoid function. The Sigmoid function is continuous and smooth. The function is shaped like an "S". When x tends to negative infinity, y is 0. When x tends to positive infinity, y is 1. It takes values in the range $[0, 1]$, corresponding to the probability range. The orange represents the Tanh function. This function is different from the Sigmoid function. It is a completely symmetric function. The range of values is $[-1, 1]$.

2.2.1.1 Step Function

The step function is the simplest activation function, see Figure 2.4. This function is usually chosen when we want to create a binary classifier. However, the step function does not meet the needs of a multi-classifier. Since the gradient of the step function is zero, it hinders the backpropagation process [53]. The mathematical expression of the step function is shown below:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & otherwise \end{cases}$$

2.2.1.2 Sigmoid Function

The sigmoid function is also known as the Logistic function. It is a widely used nonlinear activation function, see Figure 2.4. The sigmoid function takes values in the range of $(0, 1)$ and can be used for binary classification [39, 53]. The expression of the sigmoid function shows as Equation 2.1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

The sigmoid function is a continuous function with output in the range of $(0, 1)$, for which it is easy to find the derivative. However, in backpropagation, the gradient can quickly vanish, and thus the training of the deep network cannot be completed.

2.2.1.3 Tanh Function

Tanh function is also known as the hyperbolic tangent function, see Figure 2.4. It takes values in the range of $[-1, 1]$. The Tanh function is defined as Equation 2.2.

The Tanh function is continuous and differentiable. Compared to the sigmoid function, the gradient of the Tanh function is steeper, and its gradient is not restricted to vary in a specific direction, which makes the Tanh function more popular than the sigmoid function. However, it still suffers from the problem of gradient vanishing [39, 53].

$$f(x) = \tanh x = \frac{\sinh x}{\cosh x} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

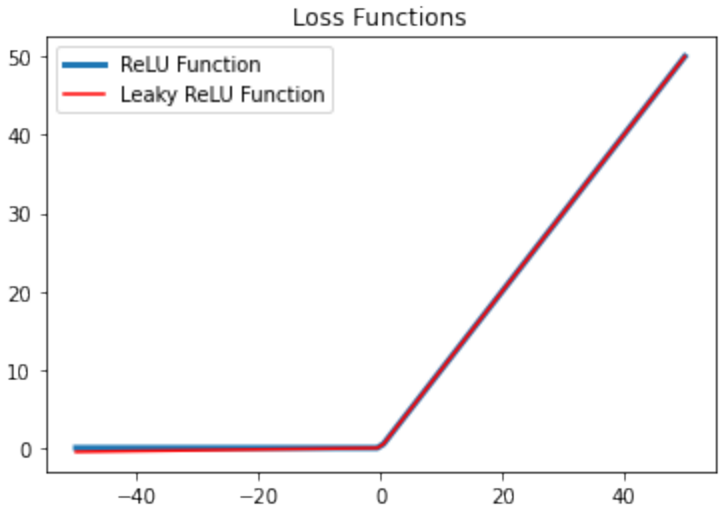

Figure 2.5: The blue line represents the ReLU function. It is a nonlinear function that can be used for backpropagation. The ReLU function has an output of 0 when the input is negative. In this way, the neurons are not activated, making the network sparse and facilitating network computation. The red line represents the Leaky ReLU function. It is designed to solve the problem that when x is less than 0, the neuron is not activated when the ReLU function is used. In the Leaky ReLU function, a slope is set so that when x is less than 0, the output is not all 0. The slope in the figure is 0.01.

2.2.1.4 ReLU Function

ReLU is a nonlinear activation function, see Figure 2.5, which is widely used in neural networks. The advantage of using the ReLU function is that all neurons are not activated simultaneously. In other words, a neuron is deactivated when the output of the linear transformation is 0. The ReLU function is defined as follows: $R(z) = max(0, z)$ [53].

2.2.1.5 Leaky ReLU Function

The Leaky ReLU function is designed to solve the dead ReLU problem, see Figure 2.5. A number of 0.01 is used to initialize the neuron so that the ReLU is more biased towards activation in the negative region rather than directly to 0 [53]. The specific mathematical expression is as follows:

$$f(x) = \begin{cases} 0.01x & x < 0 \\ x & x \geq 0 \end{cases}$$

2.2.2 Loss Function

The loss function plays a crucial role in the performance of the model. Choosing the correct loss function can help the model focus on the right set of features in the data and thus achieve optimal and faster convergence.

2.2.2.1 Pixel-wise Loss

The pixel-wise loss function calculates the inter-pixel loss between the predicted image and the ground truth. This loss function consists of L_1 loss, L_2 loss and cross-entropy loss, all of which can be applied to predict between each pair of pixels of the target variable. One of the L_1 loss functions is also called Least Absolute Error which is the sum of the absolute values of the differences between the ground truth and the predicted value (as in Eq 2.3a). The L_2 loss function is also called Least Square Error which is the sum of the squares of the differences between the ground truth and predicted values (as in Eq 2.3b).

$$loss(x, y) = \frac{1}{n}\sum_{i=1}^{n} |y_i - f(x_i)| \tag{2.3a}$$

$$loss(x, y) = \frac{1}{n}\sum_{i=1}^{n} \|y_i - f(x_i)\|^2 \tag{2.3b}$$

where i denotes the i^{th} sample, n denotes the total number of samples, x denotes the input data, y denotes the ground truth, and f denotes the mapping relationship between input and output.

For L_1 loss, its derivative concerning the loss value is a constant. In the late training period, when the loss value is small, if the learning rate is constant, the loss function will fluctuate around the stable value, and it is difficult to converge to higher accuracy. For L_2 loss, the derivative is different at each point, and the further away from the lowest point, the larger the gradient. Using the gradient descent method may lead to gradient explosion when solving, so many deep learning networks choose cross-entropy loss as the loss function of the model (e.g., Eq 2.4).

$$H(p, q) = \sum_i p_i \cdot \ln \frac{1}{q_i} = -\sum_i p_i \ln q_i \tag{2.4}$$

where i denotes the index of i^{th} sample, p denotes the true distribution and q denotes the predicted distribution, then $H(p, q)$ is called the cross-entropy.

Cross-entropy is an essential concept in Shannon's information theory, mainly used to measure the discrepancy information between two probability distributions. The Cross-entropy Loss function can measure the similarity between p and q. Because of this property of cross-entropy loss, we can use it to measure the gap between the predicted semantic segmentation results and the ground truth.

2.2.2.2 Adversarial Loss

The standard GAN loss function is also known as min-max loss. It was first proposed by Goodfellow et al. [16]. The GAN loss function is shown in Eq 2.5. The standard GAN loss function can be further divided into two parts, namely, discriminator loss and generator loss. The generator tries to minimize this function while the discriminator tries

to maximize it. In practice, this loss function is saturated for the generator. In other words, if the learning ability of the generator does not catch up with the discriminator, then the network will often stop training.

$$\min_G \max_D V(D,G) = E_{x\sim P_data(x)}[\log D(x)] + E_{z\sim P_z(z)}[\log(1 - D(G(z)))] \tag{2.5}$$

$$\bigtriangledown_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^i) + \log\big(1 - D(G(z^i))\big)] \tag{2.6}$$

$$\bigtriangledown_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\big(1 - D(G(z^i))\big) \tag{2.7}$$

where i denotes the index if i^{th} sample, m indicates the total number of samples. x represents the real samples, and z represents a noisy sample in the latent space. Here "latent space" refers to a mathematical space that maps what the neural network learns from the images in a training set [23]. G is short for generator. D is short for discriminator. $G(z)$ indicates that the generator generates false images in random noise. The whole loss function is generated at the output of D, which is generally 0 or 1.

1. Discriminator Loss: When the discriminator is trained, it classifies the real data and the fake data generated by the generator. It will be penalized if it treats real instances as false or treats false instances as real. The discriminator has to maximize the equation as in Eq 2.6 during the training process to train the model.

2. Generator Loss: When the generator is trained, it samples random noise and produces an output from that noise. The output then passes through a discriminator. The discriminator is classified as "true" or "false", depending on its ability to distinguish between true and false. The loss of the generator is then calculated from the discriminator's behaviour. If generator successfully fools the discriminator, it is rewarded. Otherwise, it is penalized. The generator has to minimize the equation as in Eq 2.7 during the training process to train the model.

2.3 Deep Learning

When machine learning began, the backpropagation algorithm [51] was proposed as one of the training methods for neural networks. At that time, the neural network was a shallow

model containing only one hidden layer. However, as the difficulty of the prediction task increased, the shallow neural network could not meet the prediction accuracy requirements.

Subsequently, Geoffrey E Hinton and Ruslan R Salakhutdinov made two main points about multi-hidden layers [19]:

1. Neural networks with multiple hidden layers have excellent feature learning capabilities. Such networks can learn more essential data features, which can be helpful for visualization or classification.

2. The difficulty of training deep neural networks can be effectively overcome by "layer-wise pre-training".

This paper [19] solves the problem of gradient vanishing in the training of deep networks by initializing the network weights with unsupervised predictive training and then fine-tuning the model with supervised learning to complete the training. This idea gives deep learning a significant advantage in complex tasks.

Deep learning is a subset of machine learning, essentially a neural network with three or more layers. The core of deep learning is feature learning, which aims to obtain hierarchical feature information through a hierarchical network, thus solving the critical problem of manually designed features. Hierarchical feature information means that as the depth of the network increases, the complexity of the extracted image features also increases. Different network models are needed to achieve better results for various problems (image, speech, text). Several popular deep learning networks such as CNN and deep generative models are discussed below [44].

2.3.1 Convolutional Neural Network

CNN is a popular and widely used algorithm in deep learning. It has been commonly used in different application areas such as natural language processing, speech processing and computer vision. With increasing computing power, some large CNN networks have shown significant advantages in many fields, especially images. In 2012, Krizhevsky et al. [24] proposed the AlexNet network architecture and won the ImageNet image classification competition. Subsequently, different scholars have proposed a series of network structures and continue to improve the performance of ImageNet, among which the classic networks include VGG, GoogLeNet and ResNet.

CNNs have the main advantage of having local connections. The network uses local connections and shared weights instead of the traditional fully connected network. Such a network is faster and easier to train due to its very few parameters. This operation is similar to the operation in visual cortical cells. These cells are sensitive to specific parts of the scene rather than the whole scene. In other words, these cells are used as local filters on the input and extract the spatially local correlations present in the data.

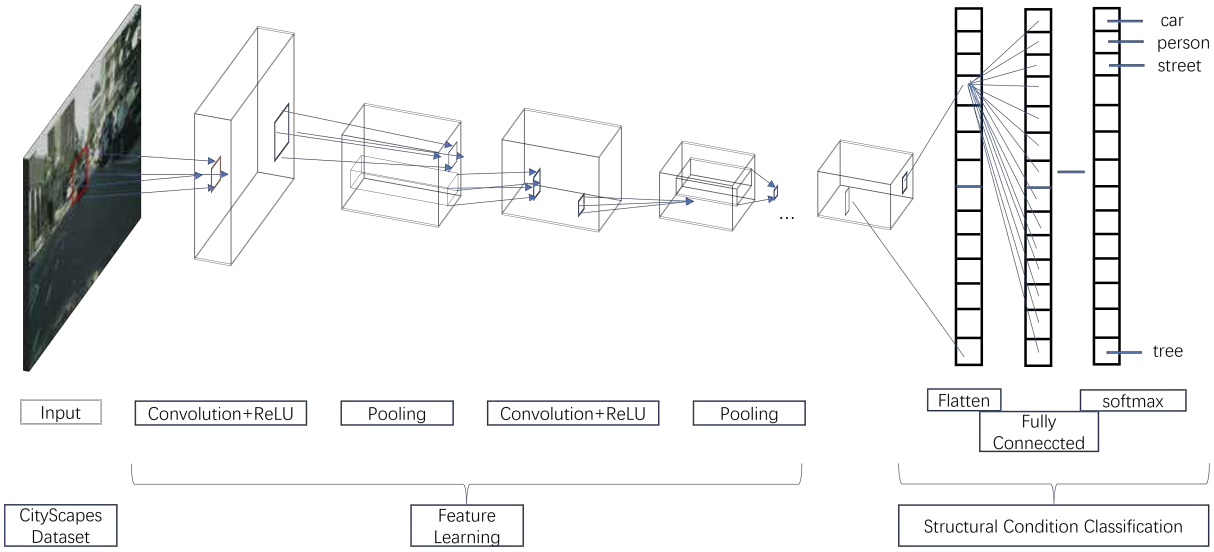

Figure 2.6: The figure shows the structure of a convolutional neural network. A convolutional neural network consists of an input layer, a convolutional layer, a ReLU layer (as an example), a pooling layer and a fully connected layer. By stacking these layers together, a complete convolutional neural network can be constructed.

A typical CNN structure consists of an input layer, a convolutional layer, a pooling layer, a fully connected layer (identical to multilayer perceptron) and an output layer. The input layer is used to receive data and pass it on to the rest of the neural network. The convolutional layer performs convolutional operations on the input data and passes the result to the next layer. Pooling layers are used to reduce the dimensions of the feature maps. They summarize the features present in a region of the feature map generated by a convolution layer. Fully connected layers are an essential component of CNNs, which connect every neuron in one layer to another. The convolutional layer, activate function and pooling layers are set alternately. The CNN structure is shown in Figure 2.6).

The input image of a CNN is arranged in three dimensions, i.e., $m \times m \times r$, where m refers to the height and width of the input image, and r refers to the depth of the image or

the number of channels (e.g., for RGB images, $r = 3$). In each convolutional layer, there are k filters of size $n \times n \times q$, also called kernels. Here n is generally smaller than the input image size. q can be smaller or of the same size as r. As mentioned before, filters are the basis for local connections, which are convolved with the input. The filters share the same parameters (weights W_i and bias b_i, $i \in [1, k]$) among themselves, generating k feature maps (h_i). Each feature map is of size $m - n - 1$. Similar to the Multilayer Perceptron, the convolution layer is used to compute the dot product between the weights and the input [44], but the input is a partial region of the original input image. An activation function or a nonlinear function f is then applied to the output of the convolution layer. In the subsampling layers of the network, each feature map is downsampled. This is done to reduce the number of parameters in the network, speed up the training process and control overfitting. The pooling operation (e.g., average or maximum) is performed in a continuous region of $p \times p$ (where p is the filter size) of all feature maps. The final stage of the network usually uses fully connected layers. These layers use the low-level and mid-level features previously extracted to generate high-level image feature information from the data. The details of an image are low-level features, such as lines or dots, as well as high-level features that are built on top of low-level features to detect objects and larger shapes in the image. CNN uses both types of features. The first couple of convolutional layers will learn filters for finding low-level features, while the later layers will learn high-level features [25]. The final layer (e.g., Softmax or SVM) is used to generate classification scores, each of which is the probability of a particular class.

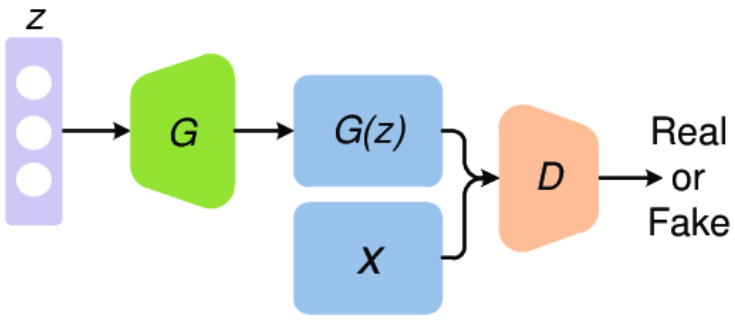

Figure 2.7: The figure shows the basic structure of a generative adversarial network network. In the figure, z is the noise, and x is the training set of the model. The generative adversarial network mainly consists of a generator G and a discriminator D. The generator generates fake images using random noise to fool the discriminator. The discriminator is used to determine the authenticity of the input images. The neural network is trained by adversarial learning. The image is from [41]. Copyright ©2019, IEEE

2.3.2 Generative Adversarial Network

GAN is an unsupervised learning method. The basic structure of GAN is shown in Figure 2.7. The Generative Network (the blue part in Figure 2.7) takes a random sample from the latent space as input. Its output needs to mimic the real sample in the training set as closely as possible. The discriminator network (the orange part in Figure 2.7) takes as input the real sample or the output of the generative network. It aims to distinguish the output of the Generative Network from the real sample as much as possible. On the other hand, the generative network tries to deceive the discriminator network as much as possible. These two networks fight against each other and constantly adjust their parameters until the discriminator network is unable to determine whether the output of the Generator Network is real or not.

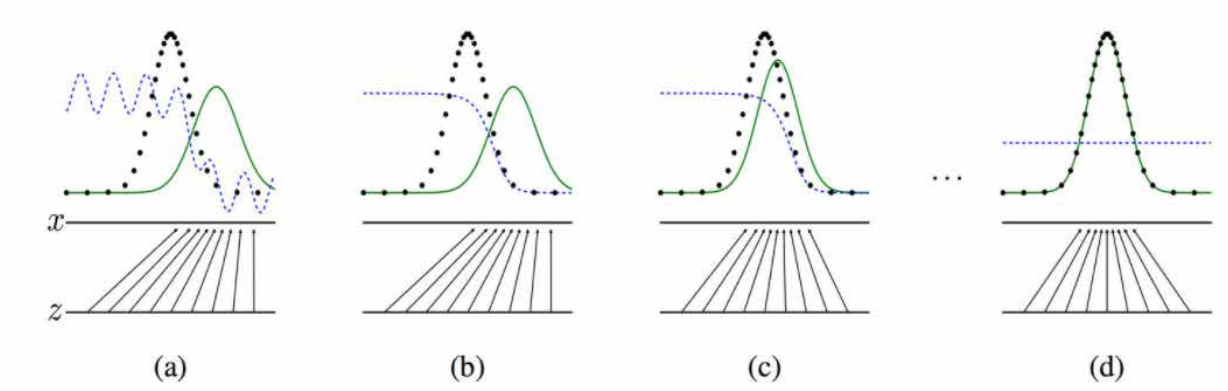

Figure 2.8: The diagram shows the generator, discriminator and sample distribution. In the figure, z represents noise, the arrows correspond to the distribution of z and x is the training set of the model. The black dashed curve in the figure indicates the real sample distribution. The blue dashed curve shows the discriminator discriminant probability distribution. The solid green curve shows the distribution of the generated samples. The sample state (a) is an initial state; the distribution generated by the generator is very different from the real sample distribution, and the discriminator is not very stable in discriminating samples. The sample state (b) shows that the discriminated samples are distinguished significantly and well. The sample state (c) shows the generator distribution approximates the real sample distribution compared to the previous one. After many iterations, the final ideal situation (d) is that the discriminator does not discriminate against the real samples. The image is from [16]. Permission is requested.

A GAN network is trained in two phases. The first phase is to fix the discriminator and

only the generator is trained. The second stage is to fix the generator and only the discriminator is trained. Figure 2.8 shows the distributions of the generator, discriminator and the samples during the training of the GAN network. From that figure, we can find that in the initial state (a), the distribution generated by the generator is very different from the real distribution, and the discriminator is not very stable in discriminating samples. Therefore, the discriminator will be trained first to discriminate the samples better. The discriminator is trained several times to reach the sample state (b) when the discriminated samples are distinguished very significantly and well. Then, the generator is trained again. After training the generator to reach the sample state (c), the generator distribution approximates the true sample distribution compared to the previous one. After several training iterations, we eventually hope to reach the state (d), where the generated sample distribution fits the real sample distribution, and the discriminator cannot distinguish whether the sample is generated or real (the discriminant probability is 0.5). This state means that we can generate samples close to the real image at this time, as seen from (d). Eventually, the GAN network learns the image distribution of the source dataset accurately. In this thesis, GAN can capture the distribution of the image set. In the later section, we will describe the GAN network in the DA model.

2.4 Deep-Learning Semantic Segmentation Models

In [35], Shervin Minae et al. group deep learning-based segmentation works into 11 categories based on their main technical contributions. We will introduce four of them as they are related to our work. Our model combines DeepLab V2 and GAN. All the methods we used are discussed in detail below.

2.4.1 Convolutional Model with Graphical Models

In order to integrate more contextual information, some approaches try to combine probabilistic graphical models such as Conditional Random Fields (CRF) and Markov Random Fields with the CNN models. Where a random field refers a random function over an arbitrary domain, it is a function that takes on a random value at each point [65].

Chen et al. [6] proposed an algorithm based on a combination of CNNs and a fully connected CRF for image semantic segmentation (shown in Figure 2.9). In that paper, they found that using the feature map from the last layer of the deep CNN for object segmentation is not sufficient since such features are not localized enough. To overcome

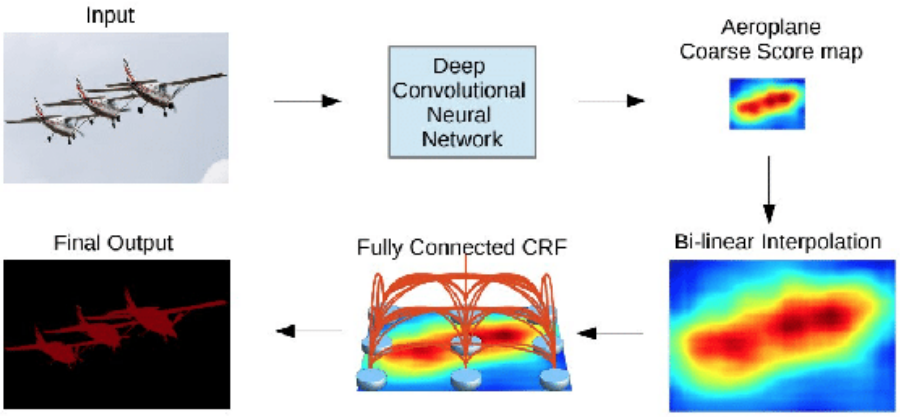

Figure 2.9: A combination of Convolutional Neural Networks and fully connected Conditional Random Fields for image semantic segmentation. A deep convolutional neural network, such as VGG-16 or ResNet-101, is used as a feature extractor. The deconvolution continuously downsamples the feature map. The feature map is then scaled to the original image resolution using a bilinear interpolation method. Then, a fully connected Conditional Random Fields is used to refine the segmentation results. The image is from [6]. Copyright ©2018, IEEE

the localization problem, they combined the response of the last layer of the CNN model with a fully connected CRF. They confirmed that this model could localize the object boundaries more accurately through experiments.

In another related work, Lin et al. [27] proposed an efficient semantic segmentation algorithm based on CRF models. In that paper, they explored the "patch–patch" context (between image regions) and the "patch–background" context. To capture the "patch–patch" context, they combine a CRF and a CNN-based pairwise model to capture semantic correlations between neighbouring patches. The authors show that a network design with traditional multi-scale image input and sliding pyramid pooling can perform well to capture the patch–background context.

2.4.2 Multi-Scale and Pyramid Network

Multiscale analysis is an idea in image processing that has now been applied in various neural network architectures. The low-level features have less semantic information in the image task, but the target location is accurate. The higher layers are rich in semantic details, but the target location is coarse.

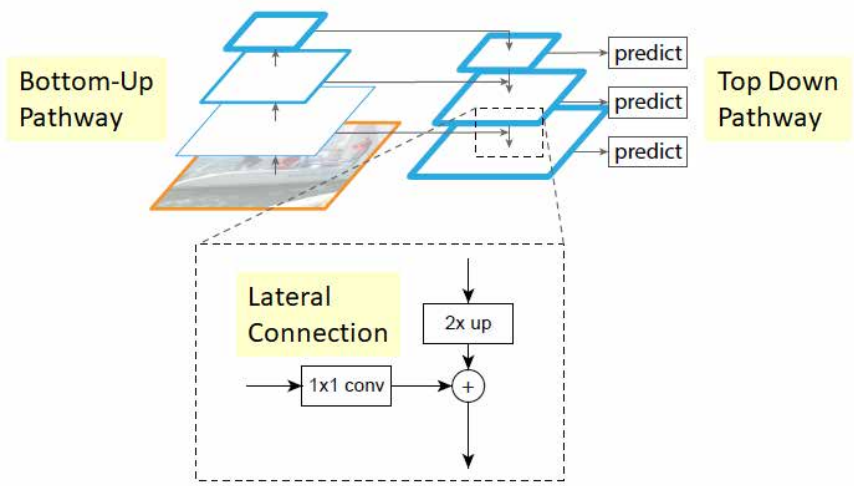

Figure 2.10: The diagram shows the structure of the feature pyramid network. The feature pyramid network adopts top-down and lateral connection methods. This allows high-resolution, low-semantic features and low-resolution, high-semantic features to be fused together. The resulting feature maps at different scales are rich in semantic information. The image is from [28]. Copyright ©2017, IEEE

To make the model good at feature capturing, Lin et al. proposed a Feature Pyramid Network (FPN) [28]. FPN is a neural network consisting of a bottom-up path, a top-down path, and lateral connections. The specific model structure is shown in Figure 2.10. The top-down path is to up-sample a higher-level feature map that is more semantic. The features are then connected laterally to the previous layer of features. The authors then use a 3×3 convolutional kernel to process the fused feature maps. Convolution generates the output of each stage. Finally, each stage of the top-down pathway generates a prediction to detect an object. Here FPN is mainly developed for target detection, but for now, it is also applied to image segmentation.

Then, Zhao et al. [17] proposed the Pyramid Scene Parsing Network. In the paper, they pointed out that for complex scenes, there are not only many objects with similar shapes, but also some overlapping objects. In this case, the baseline model (Fully Convolutional Network (FCN)) is not good at capturing the adjacency relationship between two adjacent objects. It will cause a category confusion problem. In contrast, Pyramid Scene Parsing Network can capture the contextual relationship of images. This ability is essential for semantic segmentation.

The structure of the Pyramid Scene Parsing Network model is shown in Figure 2.11, where a feature extraction network extracts the features of the input images. The extracted features are used as the pyramid pooling module's input. In the pyramid pooling module, the authors construct a feature pyramid of depth 4. The features of different depths are obtained by pooling at different scales based on the input features. The pooled feature dimensions given in the paper are 1×1, 2×2, 3×3 and 6×6. The feature dimensions are then reduced to $\frac{1}{4}$ of the original dimension by a 1×1 convolution layer. Finally, these pyramid features are upsampled directly to the same size as the input features and then concatenated with the input features to obtain the final output feature map. The merged feature is a fusion of the target's detailed features (shallow features) and global features (deep features, i.e., contextual information). In this way, the model can use the image context information for semantic segmentation.

2.4.3 Dilated Convolutional Models and DeepLab Family

Dilated convolution is also known as atrous convolution. The dilated convolutions support the exponential expansion of the receptive field without loss of resolution or coverage. That means the dilated convolution allows the original 3×3 convolution kernel to have a larger Receptive Field for the same number of parameters and computational effort. Here receptive field means the region in the input space that affects a particular CNN's feature [45].

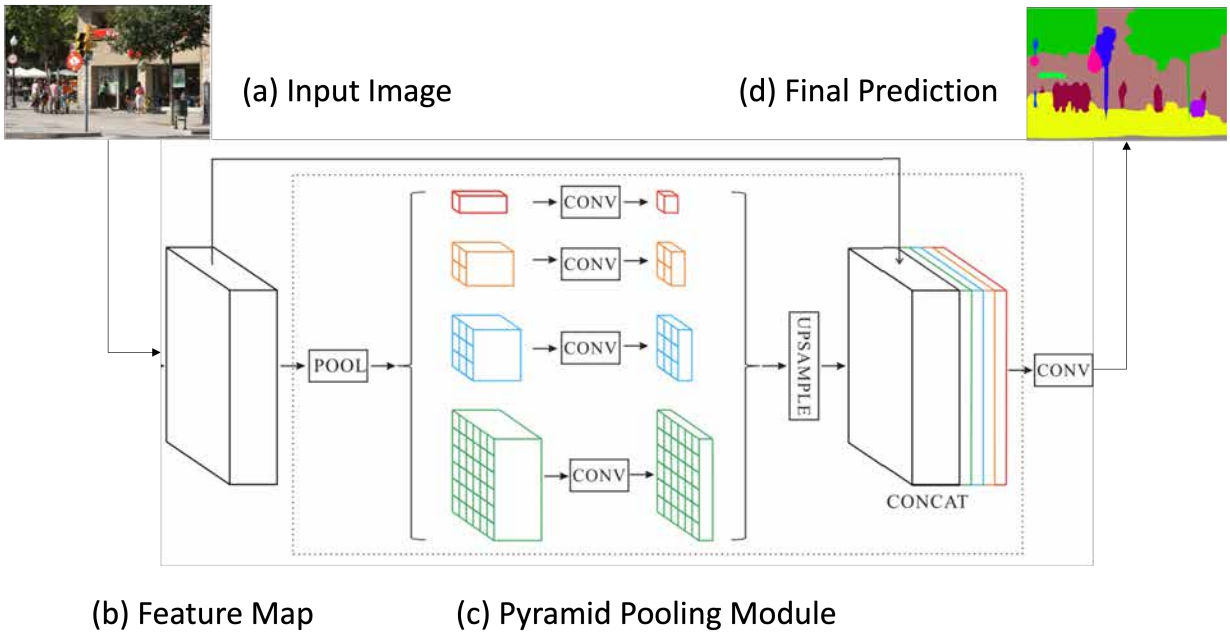

Figure 2.11: Figure shows the Pyramid Scene Parsing Network model. (a) shows the input image. (b) shows the feature map of the input image extracted using Convolutional Neural Network. Then, the feature representations of different sizes are obtained using image pyramids. The feature representations are then fused using upsampling and cascading to produce the final pixel prediction. The image is from [17]. Copyright ©2017, IEEE

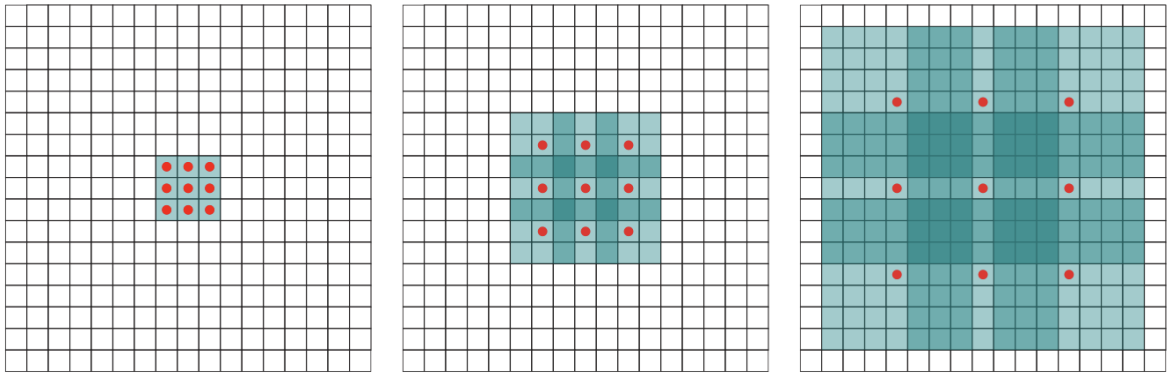

Figure 2.12: The structure of the dilated convolution model. (a) indicates that the feature map is generated by a 1-dilated convolution. The receptive field of each element in the feature map is $3 * 3$. (b) indicates that the feature map is generated by a 2-dilated convolution. The receptive field of each element in the feature map is $7 * 7$. (c) indicates that the feature map is generated by a 4-dilated convolution. The receptive field of each pixel in the feature is $15 * 15$. The image is from [2]. Copyright ©2020, IEEE

The dilated convolution operation shows in Figure 2.12. Figure 2.12 (a) is the base convolution kernel. The above figure (b) corresponds to a convolution with 3×3 and a dilation rate $= 2$. The receptive field is equivalent to a kernel size of 7×7. Figure (c) has a dilation rate $= 4$. Using this dilation rate, the receptive field is equivalent to a convolution kernel of 15×15. As the size of the convolution kernel becomes larger, the receptive field naturally becomes larger. The dilated convolution follows Eq 2.8.

$$RF_i = RF_{i-1} + (k-1) \times s \tag{2.8}$$

where RF_{i-1} is the receptive field of the previous layer, i denotes the i_{th} layer, k is the size of the convolution kernel, and s is the step size.

Dilated convolution is popular in real-time segmentation, and this technique has been utilized in many articles. The most important uses of dilated convolution include the the DeepLab series [7], multiscale background aggregation [71] and densely connected atrous spatial puzzles [64].

There are three versions of the DeepLab family, and our model mainly uses the DeepLab V2 model. The structure of the DeepLab v2 model is shown in Figure 2.13. It has three

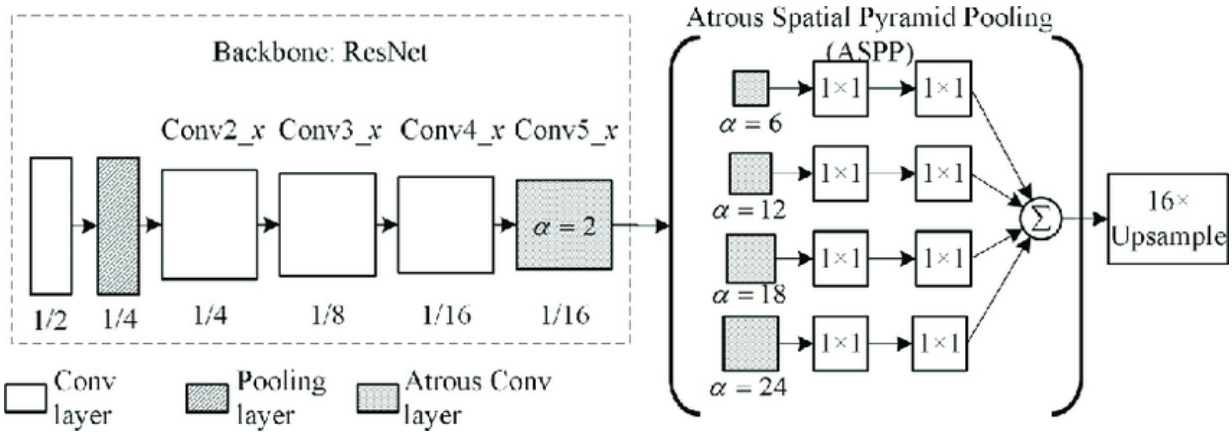

Figure 2.13: The structure of DeepLab V2. The DeepLab V2 model consists of ResNet-101 and atrous spatial pyramid pooling. Atrous spatial pyramid pooling uses the fusion of multi-scale features to improve the accuracy of segmentation results. The image is from [43]. It has open access.

outstanding features. First, the model utilizes dilated convolution for dense prediction tasks. Dilated convolution can effectively expand the receptive field. It helps the model to integrate more contextual information of the images without additional parameters and computational effort. Second, it uses atrous spatial pyramid pooling, which can fuse multi-scale feature maps to accurately capture object adjacencies. Third, it improves object boundary localization by combining methods from deep CNN and CRF. These three features make DeepLab V2 perform well in the semantic segmentation task, and that is why we chose it as the backbone.

2.4.4 Generative Models

Since being proposed, GANs have been widely used for various tasks in computer vision. In the following, we will focus on the classical model GANs used for semantic segmentation.

Luc et al. [32] first used the GAN idea for semantic segmentation. The specific network structure is shown in Fig 2.14. The segmentor on the left side of the figure is the traditional CNN-based segmentation network. Convolution and deconvolution processes can be seen in the "Convnet" part. The adversarial network on the right is the discriminator in GAN. Sigmoid activation is used at the end of the adversarial network for binary classification. The segmentor is used for class prediction and feature extraction. The discriminator determines whether the input is ground truth or the segmentor's prediction. The generator and

the discriminator improve each other's ability during the training process. This process improves the segmentation accuracy and the discriminative ability.

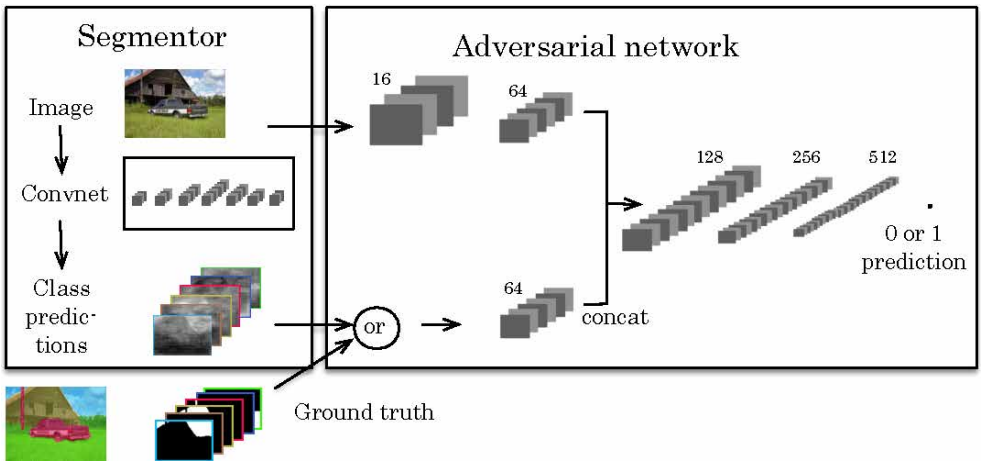

Figure 2.14: The structure of adversarial networks. The network in the box on the left is a segmentation network using RGB images as input and assigning categories to each pixel. The network in the right box represents the adversarial network. It takes the label map as input to generate category labels. (ground truth=1, otherwise=0). The image is from [32]. Permission is requested.

2.5 Domain Adaptation

DA is a representative transfer learning method that shifts data with different distributions in the source and target domains into one feature space. The main purpose of DA is to let these distributions be as close as possible to each other in the feature space. The main idea of DA is to use the model trained by source data to predict the target data. The main problem of DA is how to reduce the differences between the different distributions of the source and the target. Various methods have been proposed to solve this problem. The essence of these methods is to adjust the feature distribution between the source and target data so that the objective function learned from the source domain in the feature space can be migrated to the target domain to improve the accuracy of the target domain.

DA includes unsupervised DA and semi-supervised DA. The data in the source domain have their corresponding labels. In unsupervised DA, targets have no labels at all, and in semi-supervised DA, targets have some labels, but the number is very small.

Youshan Zhang et al. [77] proposed six different DA methods. In this thesis, we selected only four methods that are relevant to our model.

2.5.1 Discrepancy-Based Domain Adaptation

The discrepancy-based approach is one of the commonly used methods in DA models. It aims to reduce the difference between two domains while making the data distribution consistent across domains during the training process. The discrepancy-based approach can be further divided into many subgroups. We present only the commonly used methods in the following subsections.

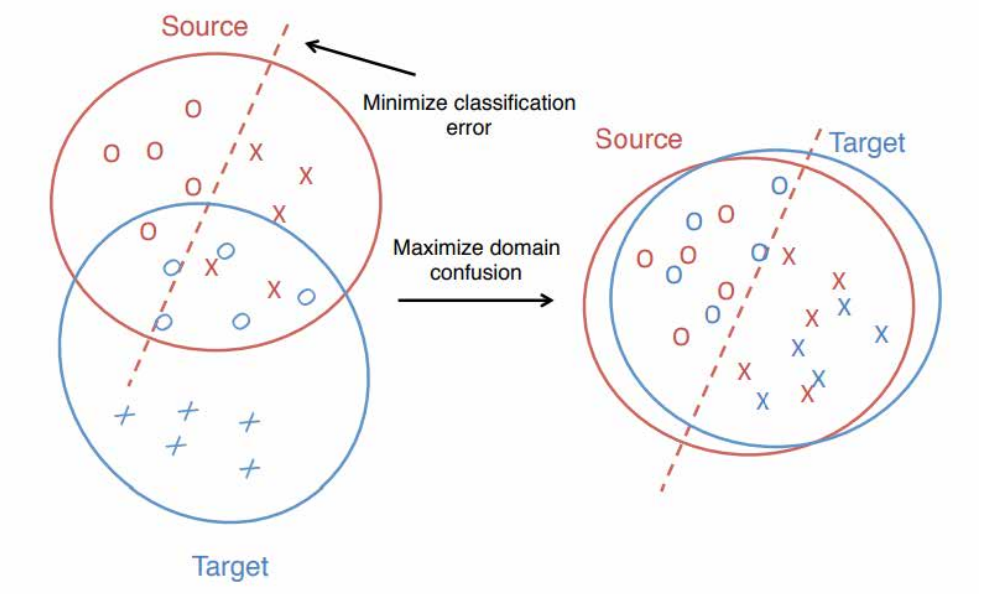

Figure 2.15: Maximum Mean Discrepancy. A model learned using source domain data is not necessarily shift well to the target domain. The representations with domain-independent distortion are learned by minimizing the classification error while maximizing domain confusion. The image is from [59]. Permission is requested.

2.5.1.1 Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) is one of the methods for minimizing the distance between two distributions. It is calculated by Equation 2.9. This equation means to

find a mapping function f. This mapping function can map the distributions of source and target domain data to a higher-dimensional space. The difference between the expectations of the random variables of the two distributions after mapping is the mean discrepancy. Finally, this function is used to maximize the mean discrepancy. This maximum value is called MMD. Some neural networks use the following approach to reduce the inter-domain variance. They choose a CNN with a fixed and robust representation capability, use MMD to decide which layer to use to minimize the inter-domain variance, and then use the reduced variance feature representation to do the classification. However, this approach does not allow convolutional networks to maximize their learning capabilities.

$$MMD(D_S, D_T) = sup_{f \in H} \|E_{X_S}[f(X_S)] - E_{X_T}[f(X_T)]\|_H^2 \tag{2.9}$$

where D_S and D_T indicate the distribution of source and target domain data, *sup* means supremum, f indicates a mapping function, E is Euler number, and H represents Hilbert space. X_S and X_T indicate source and target domain data.

The classical model of the DA method based on Equation 2.9 as its loss function is called Deep Domain Confusion which is described in [59]. The starting point of Deep Domain Confusion (as in Fig 2.15) is to reduce the difference between the source and target distributions during training to allow the network to learn a representation that is both semantically meaningful and domain invariant. The figure shows that reducing the difference makes the classifier applicable to both domains.

In fact, feature extraction and classification capability can be optimized simultaneously. Therefore, the Deep Domain Confusion approach is to add an additional adaptation layer and domain error loss to the neural network to automatically learn some feature expressions. The main use of MMD, assuming that the mapping function is ϕ, can adopt the following optimization objectives: first, to optimize the supervised classifier, and second, to reduce the difference between domains (as in Eq 2.10).

$$L = L_C(X_S, y_S) + \lambda MMD^2(X_S, X_T) \tag{2.10}$$

where $L_C(X_S, y_S)$ is used to calculate the cross-entropy loss between the existing label data (X_S) and the ground truth labels (y_S). $MMD^2(XS, XT)$ is used to calculate the distance between X_S and X_T. The hyperparameter λ determines the robustness of the confusion domain.

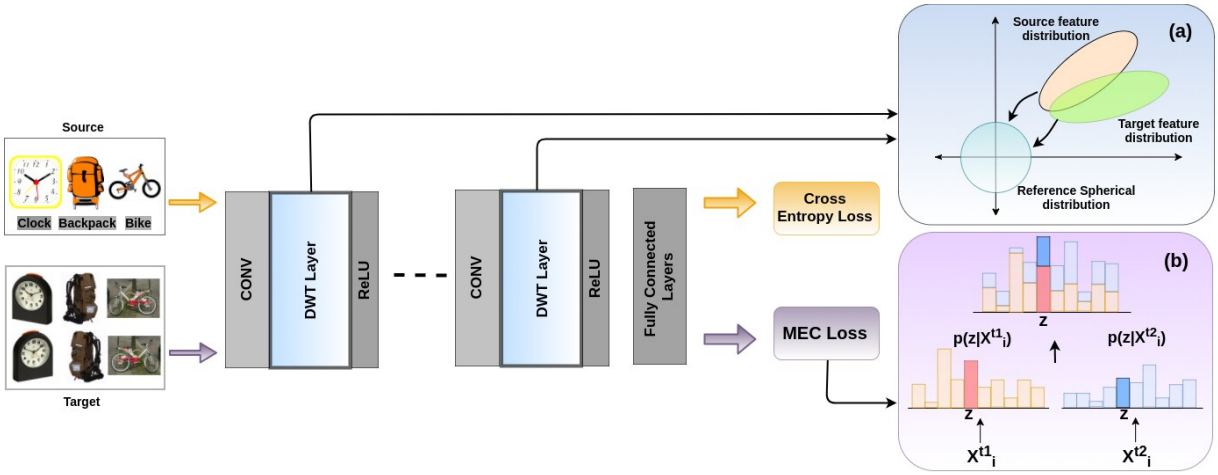

Figure 2.16: The structure of Unsupervised Domain Adaptation using feature-whitening and consensus loss. Here feature-whitening is a technique for image information correlation reduction as well as speeding up training of this deep neural network. The model is trained using the proposed Domain-specific Whitening Transform layer combined with Cross-entry loss and minimum-entropy consensus loss by Subhankar Roy et al. The image is from [49]. Copyright ©2019, IEEE

2.5.1.2 Entropy Minimization

The main objective of entropy minimization is to find the minimum entropy value between two sample distributions. In Feature Transfer Network [54], Kihyuk Sohn et al. used the entropy minimization loss function. Its role is to separate the source and target domains. The discriminative power of the Feature Transfer Network for target domain images is enhanced by optimizing the loss function. Then, Roy et al. in [49] proposed a new method for helping the model (shown in Fig 2.16) for DA, which is called Minimum Entropy Consensus. The formula is shown in Equation 2.11.

$$L^t(B_1^t, B_2^t) = \frac{1}{m}\sum_{i=1}^{m} l^t(x_i^{t_1}, x_i^{t_2}) \tag{2.11}$$

where $l^t(x_i^{t_1}, x_i^{t_2}) = -\frac{1}{2} max_{y \in Y}(logp(y|x_i^{t_1})) + logp(y|x_i^{t_2}))$. $p(y|x_i^{t_1})$ and $p(y|x_i^{t_2})$ are the probability prediction assigned by the network to $x_i^{t_2}$ and $x_i^{t_2}$ with respect to their ground-truth label y. $x_i^{t_1} \in B_1^t$, $x_i^{t_2} \in B_2^t$ and B_1^t, B_2^t are two different target batches.

2.5.1.3 Batch Normalization

Batch Normalization (BN) was proposed to address the challenge of deep model training. In model training, batch normalization utilizes the mean and standard deviation on small batches. By continuously adjusting the neural network parameters, the intermediate output of each layer of the whole neural network is made more stable.

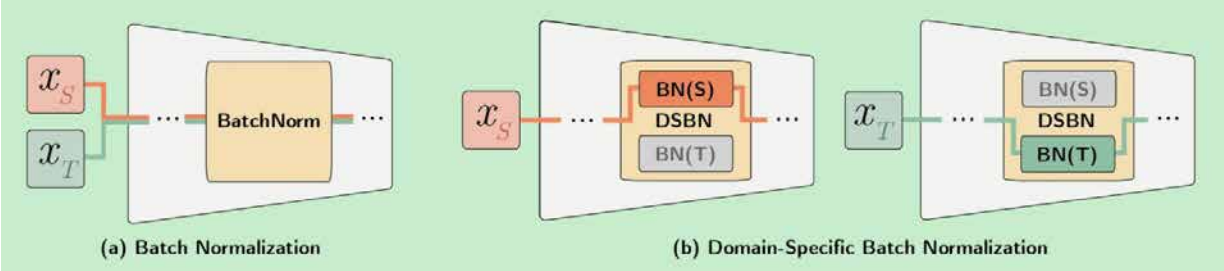

Figure 2.17: Domain-Specific Batch Normalization for Unsupervised Domain Adaptation. The difference between Batch Normalization and Domain-Specific Batch Normalization is compared. Domain-Specific Batch Normalization has two Batch Normalization branches. One branch is responsible for processing the source domain data. The other branch is responsible for processing the target domain data. The Domain-Specific Batch Normalization layer can be inserted into any unsupervised domain adaptive network with a Batch Normalization layer. The image is from [4]. Copyright ©2019, IEEE

Li et al. proposed Adaptive Batch Normalization in [26]. This method can improve the generalization ability of the model. Compared with other domain adaptive models. Adaptive Batch Normalization does not need to add additional modules or additional parameters. The Adaptive Batch Normalization method can change the parameters of the BN layer in the target domain by learning the source domain dataset. It can also update the weights of DA.

Subsequently Change et al [4] proposed a Domain-Specific Batch Normalization based on the original BN method, whose structure is shown in Figure 2.17. It works by estimating the mean and variance over multiple sets of BN layers. Additionally, domain-specific batch normalization can capture domain features. Compared to BN, it can better extract domain invariant features and is more conducive to DA of the model.

2.5.2 Adversarial-Based Domain Adaptation

Adversarial-based approaches have recently gained popularity. The model is usually combined with a GAN network and typically contains a feature extractor and a domain discriminator. The feature extractor is used to learn domain invariant features to deceive the discriminator. The discriminator is used to distinguish between the source and target domains. The differences between domains are reduced by using adversarial training.

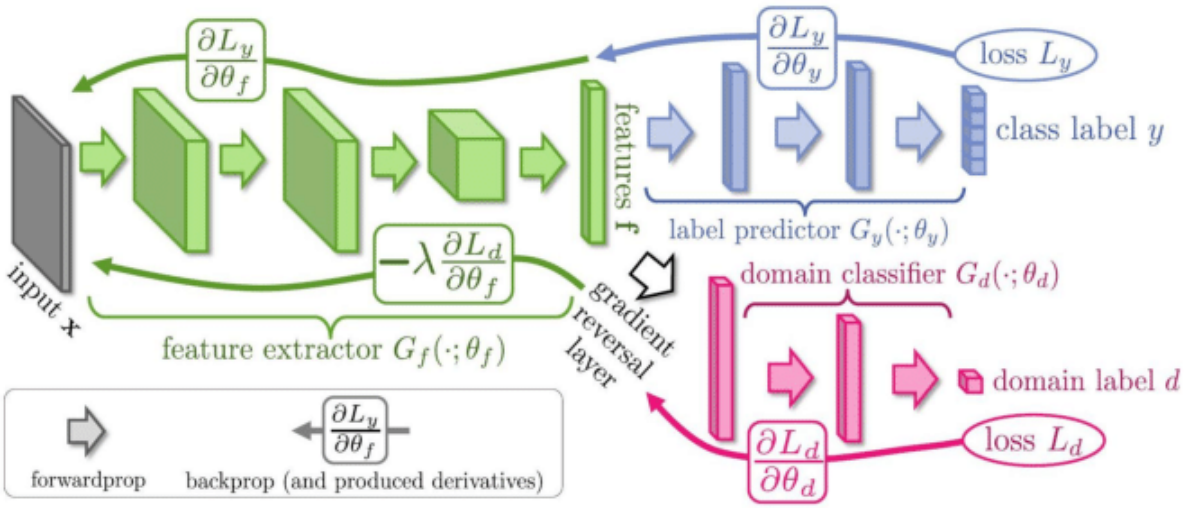

Figure 2.18: The structure of Domain-Adversarial Training. The model includes a feature extractor (green) and a label predictor (blue). Unsupervised Domain Adaptation is achieved by adding a domain classifier (red). The image is from [14]. Permission is requested.

In 2015, Ganin et al. proposed a DANN framework [14], which combines adversarial learning with DA for domain shift. This framework is outstanding in image classification tasks. The structure of DANN is shown in Fig 2.18. It consists of three main components: the feature extractor, shown in green, is used to shift the data to a specific feature space so that the label predictor can distinguish the class of data from the source domain. In contrast, the domain discriminator cannot distinguish where the data comes from. The label predictor, shown in blue, classifies the data from the source domain and classifies the correct label as much as possible. The domain classifier, shown in red, classifies the data in the feature space. It tries its best to classify whether the data comes from the source domain or target domain. In this case, the feature extractor and the label classifier form a feed-forward neural network. Then, a domain discriminator is added after the

feature extractor, connected by a gradient reversal layer. During the training process, the network continuously minimizes the loss of the label predictor for labelled data from the source domain. For all data from the source and target domains, the network continuously minimizes the loss of the domain classifier. In the DANN, the optimized objective function has two weighted components: classifier loss in the source domain and discriminator loss in the source and target domains.

In the model called Adversarial Discriminative Domain Adaptation [58] proposed by Hoffman et al., the two parts of the loss function weighted in one function do not fully account for the discriminator loss. So the classifier loss and discriminator loss are trained separately. Adversarial Discriminative Domain Adaptation can achieve higher accuracy than other classification models, and the model also provides a unified framework for later adversarial learning-based DA, see Figure 2.19.

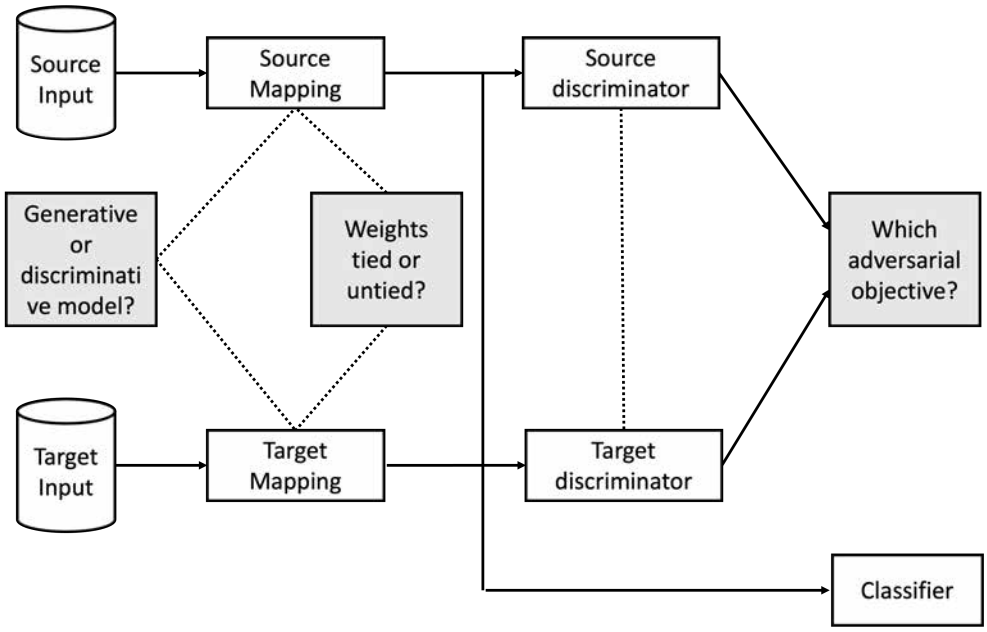

Figure 2.19: The image shows the generalized architecture for adversarial Domain Adaptation. Existing adversarial adaptation methods can be viewed as instantiations of this framework with different choices regarding their properties.

2.5.3 Pseudo-Labelling-Based Domain Adaptation

Pseudo labelling is a technique used to solve UDA. The essence of the pseudo labelling technique is to generate pseudo labels for the target based on the predicted class probabilities [5, 52, 67, 74]. Classifiers are deep networks learned from source domain data, then generate pseudo labels for the target domain. After generation, the network can use the pseudo labels as real labels. In this mechanism, different algorithms are proposed to facilitate the alignment of the distribution between two domains.

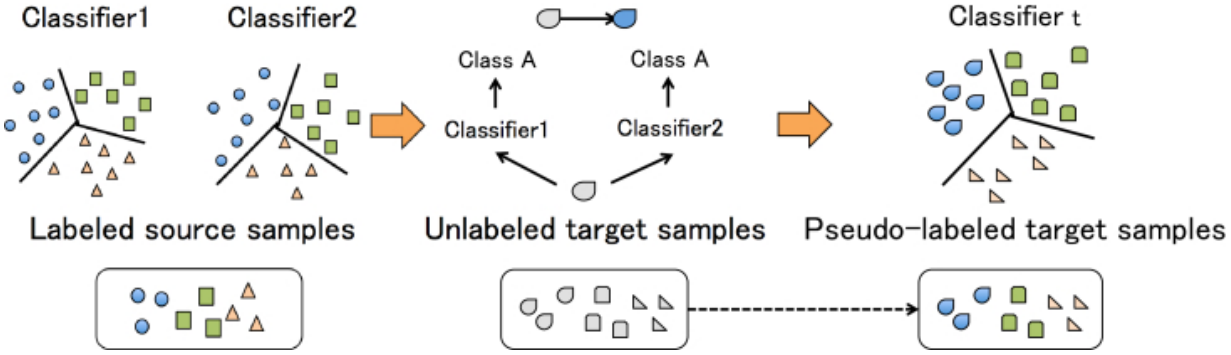

Figure 2.20: The figure shows the structure of Asymmetric Tri-training for Unsupervised Domain Adaptation. There are two classifiers $classifier_1$ and $classifier_2$. Both classifiers first predict the samples. The unlabelled target samples are assigned pseudo labels using the obtained prediction results. The image is from [52]. Permission is requested.

Saito et al. [52] proposed an Asymmetric Tri-training for UDA; the structure is shown in Figure 2.20. It works by training two classifiers with the source domain data with labels. The trained classifier in the source domain is used to label the data in the target domain. The labelling is considered reliable only if the two classifiers predict the same label or at least one of the classifiers predicts a result greater than a predefined threshold. Then, a new classifier is trained with the pseudo-labelled target domain data to obtain a discriminative representation of the target.

Chang et al. [4] proposed to combine a UDA algorithm with domain-specific batch normalization to estimate the pseudo label of samples in the target domain. Such an approach can learn domain-specific features more efficiently.

2.5.4 Reconstruction-Based Domain Adaptation

The reconstruction-based approach is to reconstruct the sample of the domain. In this way, the model can provide a good representation of the domain and, at the same time, retain domain-specific features.

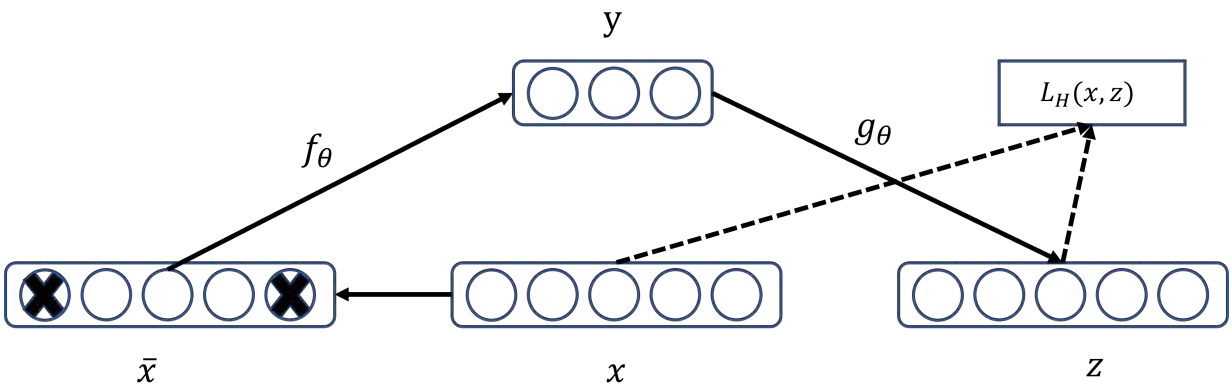

Figure 2.21: In the figure, $\widetilde{x}$ is the corrupted result of x. The autoencoder then maps it to y and tries to reconstruct x.

Self-encoders are typically used as a representation learning method for deep learning models. It maps the input into the feature space and then maps it back to the input space for reconstruction. Structurally, it consists of an encoder and a decoder. The encoder is used to extract the features from the input data, and the decoder is used to reconstruct the input data based on the extracted features.

Based on the above theory, Bengio et al. proposed a method called "corrupted" in [60]. It is performed by adding noise parameters to the traditional autoencoder, and the structure is shown in Figure 2.21. When feeding data into the model, the input is set to zero with a certain probability. The input data are then reconstructed according to the computed output data. Stacking denoising autoencoders aims to find common features between the source and target domains by an autoencoder. Its objective function is defined as Eq 2.12.

$$\theta^*\theta'^* \arg\min_{\theta^*\theta'^*} \frac{1}{n}\sum_{i=1}^{n} L(x^i, z^i) = \theta^*\theta'^* \arg\min_{\theta^*\theta'^*} \frac{1}{n}\sum_{i=1}^{n} L(x^i, g_{\theta'}(f_\theta(x^{(i)}))) \tag{2.12}$$

where x is the input data and L is the loss function. $L(x,z) = ||x-z||^2$, θ is the parameter of the autoencoder. f and g are the mapping functions.

Subsequently, K. Bousmalis et al. [1] proposed the domain separation network as shown in Figure 2.22. This network introduces the concept of a private subspace for each domain. It helps the model capture the common features between two domains by using autoencoders and loss functions. The loss function is defined as:

$$L = L_{task} + \alpha L_{recon} + \beta L_{difference} + \gamma L_{similarity} \tag{2.13}$$

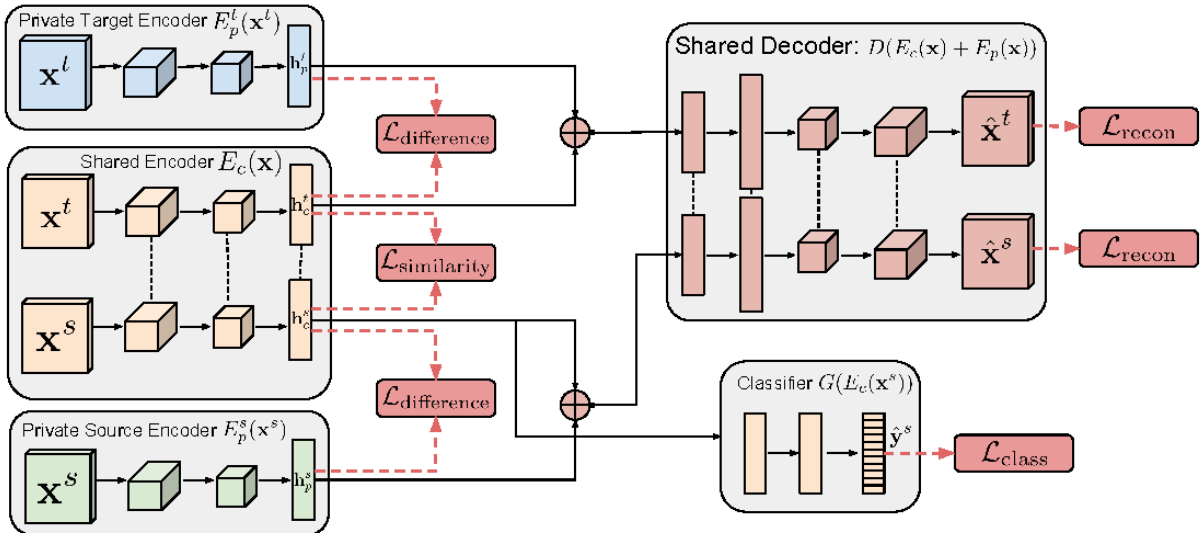

Figure 2.22: The figure shows the structure of domain separation networks. The yellow part of the figure shows an encoder $E_c(x)$ with shared weights. It is used to capture the features of a given input sample. The input data are reconstructed using two loss functions, $L_{difference}$ and $L_{similarity}$. The image is from [1]. Permission is requested.

where L_{task} denotes the loss of model training, and L_{recon} indicates the loss of data reconstruction. $L_{difference}$ is used to calculate the difference between the public space and private space. $L_{similarity}$ is used to measure the public space similarity between the source and target domains.

2.6 Mixing Datasets

Since the distributions of source and target domain images are very different, semantic segmentation by simply using the previously mentioned methods does not yield good results.

So we tried to use the image fusion method to combine the images of the source domain and target domain. This method can make the image distribution of the target domain closer to the source domain. It also allows image enhancement to make the model more robust for good inter-domain migration learning. In the following section, we present and compare some recent image fusion methods.

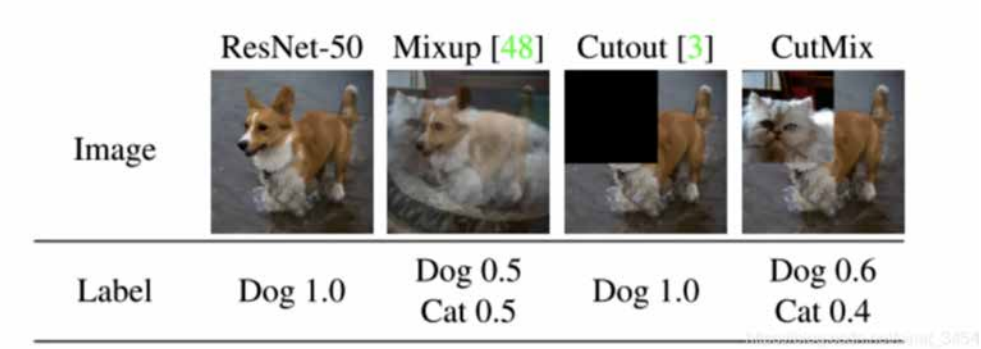

Figure 2.23: Mixing methods. Comparison of experimental results of Mixup, Cutout and CutMix methods on ImageNet and Pascal VOC 07 datasets. The image is from [72]. Copyright ©2019, IEEE

The first image fusion method is Mixup [73]. This method randomly selects two samples and mixes them. Then, the classification results of these samples also are mixed. The method of image fusion is to add each pixel value corresponding to the two images; the formula is shown in Equation 2.14. The mixing result is then passed to the model to obtain the output. The authors calculate the loss function separately for the labels of the two images and then weigh the sum of the loss functions according to the fusion ratio *lam* (shown in equation 2.15).

$$img = lam * X + (1 - lam) * X_{random} \tag{2.14}$$

where X is the image from the training dataset and X_{random} is a random-selected image from the same dataset. Parameter lam is the image fusion ratio, a random real number between [0,1].

$$loss = lam * criterion(outputs, targets_A) + (1 - lam) * criterion(outputs, targets_B) \tag{2.15}$$

where *criterion* is a function that calculates the difference between the predicted result

and the ground truth, $targets_A$ and $targets_B$ are two images from the same dataset, and *outputs* are the model prediction results.

The second image fusion method is Cutout [11] which is a method that initializes a mask M with 1s while randomly selecting a rectangular box and setting all the values in that box as 0. In order to apply the Cutout method to the semantic segmentation task, the input pixels are masked with M, thus ignoring the consistency loss of pixels masked as 0 by M. As a more robust data enhancement method, Cutout can be applied to the teacher network to generate a pseudo-label for the original image, which can train the student network. Here, the teacher network is a large complex model that distils its knowledge and passes it to train a smaller network called the student network to match the output [46]. The loss function is shown in Equation 2.16.

$$L_{cons} = \|M \odot (f_\theta(M \odot x)) - g_\phi(x)\|^2 \tag{2.16}$$

where M is an initialization mask, f_θ is the student network, and g_ϕ is the teacher network.

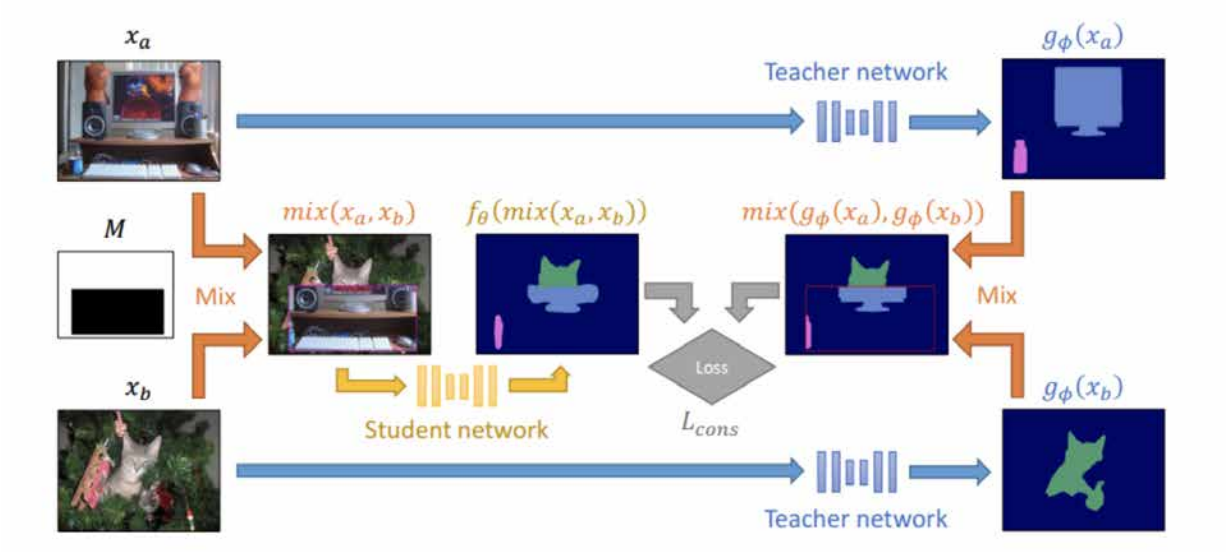

Figure 2.24: The figure illustrates the mixing regularization for semi-supervised semantic segmentation with the mean teacher framework. The model uses image fusion to combine the input images x_a and x_b. This helps the model to learn the semantic information in different images. The image is from [13]. Permission is requested.

The third image fusion method is CutMix [72].This approach requires two input images, which can be represented as x_a and x_b, to be mixed with a mask M. The authors mix

the predictions of the teacher network for these two inputs $g_\phi(x_a)$, $g_\phi(x_b)$, thus generating a pseudo-label, which can be used to monitor the predictions of the student network for the mixed images. The mix operation can be expressed as Equation 2.17. In 2019 Geoff French. et al. proposed a semi-supervised learning model (Figure 2.24), using the CutMix method for image enhancement. The specific enhancement method is shown in Figure 2.23.

Image enhancement in these three ways is more beneficial for training the neural network. The table compares the effect of the Cutout and CutMix methods on the experimental accuracy in the experimental section, as shown in Table 2.1. From that table, we can see that in semi-supervised learning, the CutMix method can be the model to obtain higher accuracy.

Table 2.1: Performance (mIoU) on CITYSCAPES validation set, presented as mean ± std-dev. The table is from [12]. Permission is requested.

Labelled samples	1/30 (100)	1/8 (372)	1/4 (744)	All (2975)
Test Group 1	Results below are from [22] and [37] with ImageNet pre-trained DeepLab V2.			
Baseline	—	56.2%	60.2%	66.0%
Adversarial [22]	—	57.1%	60.5%	66.2%
s4GAN [37]	—	59.3%	61.9%	65.8%
Test Group 2	Results below are from [72] and [37]. Same ImageNet pretrained DeepLab v2 network			
Baseline	44.41% ±1.11	55.25% ±0.66	60.57% ±1.13	67.53% ±0.35
Cutout [11]	47.21% ±1.74	57.72% ±0.83	61.96% ±0.99	67.47% ±0.68
CutMix [72]	***51.20%*** ±2.29	***60.34%*** ±1.24	***63.87%*** ±0.71	***67.68%*** ±0.37

$$mix(a, b, M) = (1 - M) \odot a + M \odot b \tag{2.17a}$$

$$L_{cons} = \|mix(g_\phi(x_a), g_\phi(x_b), M) - f_\theta(mix(x_a, x_b, M))\|^2 \tag{2.17b}$$

where M is a mask, f_θ is the student network, and g_ϕ is the teacher network, a and b are samples from the training dataset.

The forth image fusion method is proposed by Viktor Olsson et al. named ClassMix [40]. This paper introduces a new semantic segmentation strategy, using consistent regularization and pseudo-labelling for semantic segmentation. The implementation is shown in Figure 2.25, where two images A and B are sampled in unlabelled data. The prediction of these two images is performed separately using a segmentation model, and the prediction result of A is generated by the argmax function. Then, half of the predicted

Table 2.2: Performance (mIoU) on Cityscapes validation set averaged over three runs. All models use the same DeepLab-v2 network with ResNet-101 backbone. The table is from [40]. Copyright ©2021, IEEE

Labelled samples	1/30	1/8	1/4	1/2	Full(2975)
Baseline	-	55.5%	59.9%	64.1%	66.4%
Adversarial [22]	-	58.8%	62.3%	65.7%	-
Improvement	-	3.3	2.4	1.6	-
Baseline	-	56.2%	60.2%	-	66.0%
s4GAN [37]	-	59.3%	61.9%	-	65.8%
Improvement	-	3.1	1.7	-	-0.2
Baseline	44.41%	55.25%	60.57%	-	67.53%
French et al. [13]	51.20%	60.34%	63.87%	-	-
Improvement	6.79	5.09	3.3	-	-
Baseline	45.5%	56.7%	61.1%	-	66.9%
DST-CBC [12]	48.7%	60.5%	64.4%	-	-
Improvement	3.2	3.8	3.3	-	-
Baseline	43.84% ±0.71	54.84% ±1.14	60.08% ±0.62	63.02% ±0.14	66.19% ±0.11
Theirs	54.07% ±1.61	61.35% ±0.62	63.63% ±0.33	66.29% ±0.47	-
Improvement	10.23	6.51	3.55	3.27	-

labels are randomly selected as a mask, and the mask region in image A is merged into image B. The authors compare their model with the previous model, and we can see from the result Table 2.2 that the ClassMix method has a more significant improvement in the model accuracy. Due to the usefulness of the ClassMix method, Wilhelm Tranheden et al. modified it and used it in their DA model DACS [55] proposed in 2020. ClassMix, as a way of data augmentation, allows DACS to reach 52.14% accuracy.

Inspired by these papers, we tried to apply this data augmentation approach to our model and achieved great improvement. We will describe how we applied the mixing method in the following chapter.

2.7 Semantic Segmentation in Domain Adaptation

In this section, we present some previous works in the related field and analyze their strengths and weaknesses in order to compare our own model with previous models in the experimental part of this thesis.

In 2016, Judy Hoffman et al. [21] proposed a framework for unsupervised domain adaptive semantic segmentation based on fully convolutional networks [31]. Its structure is

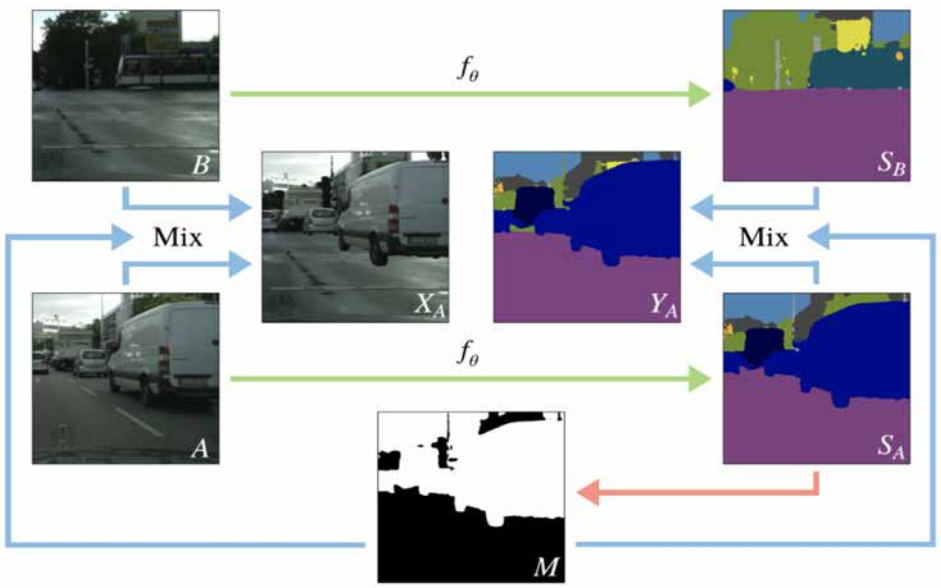

Figure 2.25: The ClassMix enhancement technique is to select two images from the unlabelled dataset. If we predict S_A based on image A and generate binary mask M, M will be used to mix image A and image B and generate a new image with its label. The image is from [40]. Copyright ©2021, IEEE

Algorithm 2.2 ClassMix algorithm Copyright ©2021, IEEE

Input: Two unlabelled samples A and B, segmentation network $f_{\theta'}$

$S_A \leftarrow f_{\theta'}(A)$

$S_B \leftarrow f_{\theta'}(B)$

$\tilde{S_A} \leftarrow arg \quad max_{c'} S_A(i,j,c') \triangleright Take\ pixel-wise\ argmax\ over\ classes$

$C \leftarrow Set\ of\ the\ different\ classes\ present\ in\ \tilde{S_A}$

$c \leftarrow Randomly\ selected\ subset\ of\ C\ such\ that\ |c| = |C|/2$

$For\ all\ i,j : M(i,j) = \begin{cases} 1, & \text{if } \tilde{S_A} \in c \\ 0, & \text{otherwise} \end{cases} \triangleright Create\ binary\ marsk.$

$X_A \leftarrow M \odot A + (1-M) \odot B \triangleright Mix\ images$

$Y_A \leftarrow M \odot S_A + (1-M) \odot S_B \triangleright Mix\ Predictions$

return X_A, Y_A

characterized by its ability to perform dense predictions without fully connected layers [21]. The proposed structure can accept input images of arbitrary size, so it is widely used for semantic segmentation. This was the first time that DA methods were applied to the semantic segmentation task of images.

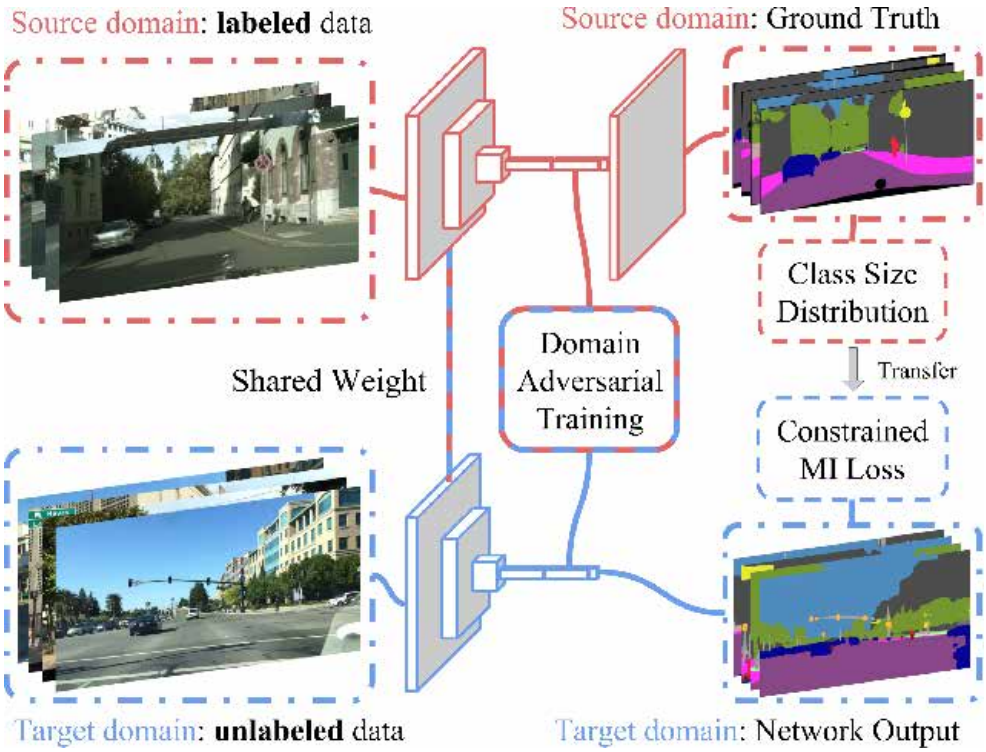

Figure 2.26: The source domain contains images with labels, and the target domain contains images without labels. The model predicts the images in the target domain by learning the features of images in the source domain. The image is from [21]. Permission is requested.

The structure of the model built using the fully convolutional network is shown in Fig 2.26. The model has two main tasks. Firstly, global features of images in different domains are aligned. Secondly, specific category features are aligned. These two tasks serve to perform DA. The model can then segment the target domain image using the features learned from the source domain. Global domain alignment is the feature extraction of images in both the source and target domains using an encoder network. Then, the image features are used for adversarial training. As in Figure 2.26, domain adversarial training, this part is like the adversarial training of GAN so that the extracted features of different

domains have a similar distribution to the complete global information. Category-specific Adaptation is the alignment of local information in the category space using Multiple Instance Learning Loss [42] with full convolutional constraints. Multiple Instance Learning Loss is used to calculate the loss of different classes. The constraints here are mainly in two aspects. One is the constraint on the presence or absence of objects in the image. Another is the constraint on the size of objects. These constraints are used to align the information on specific classes.

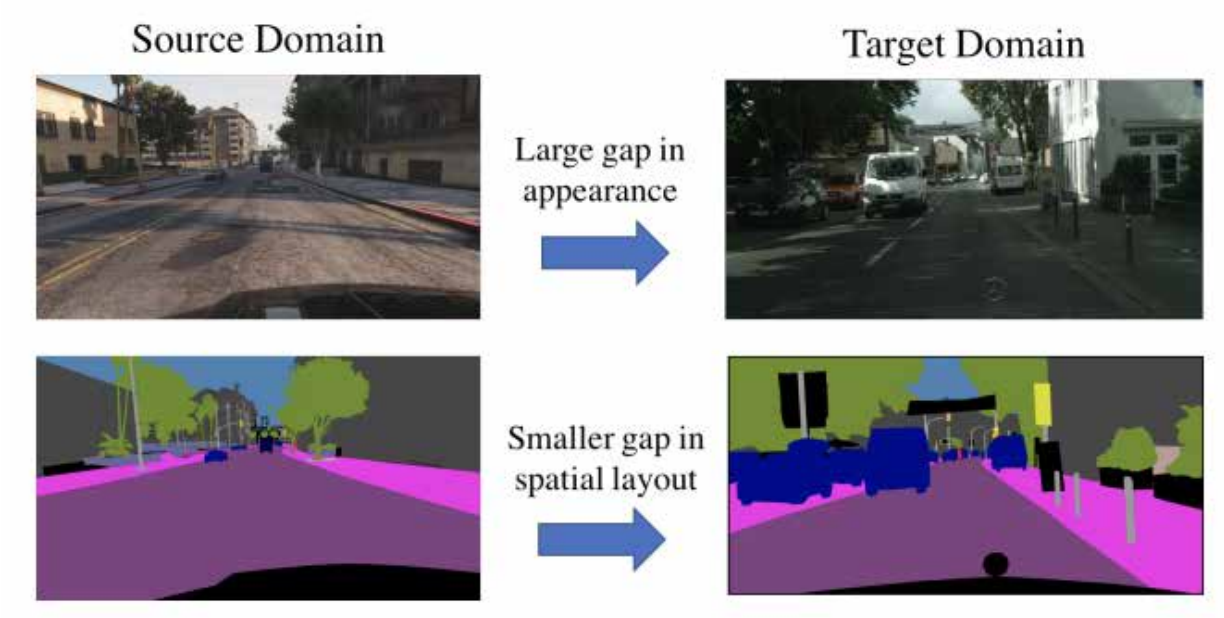

Figure 2.27: The figure shows the main idea of the AdaptSegNet model. Even though the two street views themselves are very different, they have a lot of similarities in the output space, such as spatial layout and local context. The image is from [56]. Copyright ©2018, IEEE

The experimental results of this model are shown in Table 2.3 and Table 2.4. From these tables, we can see that the model's accuracy is 27.1 for the experiment with the GTA5 dataset as the source domain and the CityScapes dataset as the target domain. The model's accuracy is 17.0 for the experiment with SYNTHIA dataset as the source domain and the CityScapes dataset as the target domain. Although the accuracy of the model mentioned in the paper [21] is not high, the model still has a milestone significance.

Then, in 2018, Yi-Hsuan Tsai et al. further explored the domain of DA [56]. They proposed a DA method for pixel-level semantic segmentation called AdaptSegNet. The

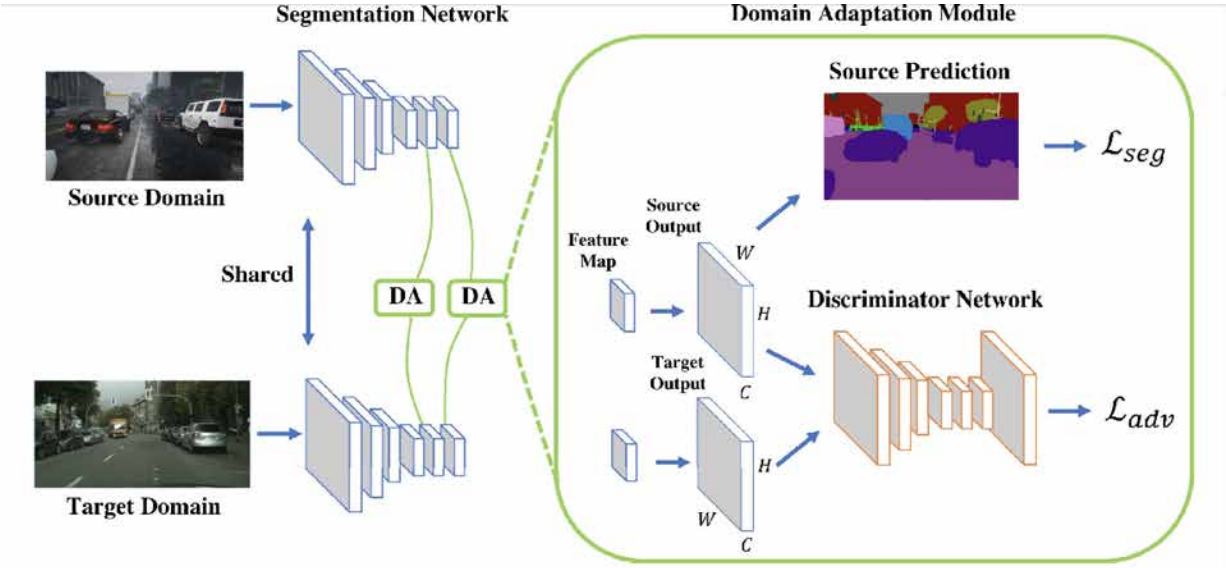

Figure 2.28: AdaptSegNet model. The training process is that the model uses the features learned from the source domain to predict the target domain images, where the source domain images are labelled, and the target domain images are unlabelled. The model performs semantic segmentation on the source and target images separately and obtains the feature maps. The feature maps are then fed into the discriminator. The discriminator is used to determine the origin of the feature map. The image is from [56]. Copyright ©2018, IEEE

Table 2.3: Adaptation from synthetic to real. Judy Hoffman et al. [21] use GTA5 as source labelled training data and CityScapes train as an unlabelled target domain. Here GA represents global domain alignment, and CA indicates category-specific adaptation. Table is from [21]. Permission is requested.

GTA5 → CityScapes																					
Method	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIou
Dialation Frontend [71]	31.9	18.9	47.7	7.4	3.1	***16.0***	10.4	1.0	76.5	13.0	58.9	36.0	1.0	67.1	***9.5***	3.7	0.0	0.0	0.0	21.1
Their Method (GA only)	67.4	29.2	***64.9***	***15.6***	***8.4***	12.4	9.8	2.7	74.1	12.8	***66.8***	38.1	2.3	63.0	9.4	5.1	0.0	3.5	0.0	25.5
Their Method (GA + CA)	***70.4***	***32.4***	62.1	14.9	5.4	10.9	***14.2***	***2.7***	***79.2***	***21.3***	64.6	***44.1***	***4.2***	***70.4***	8.0	***7.3***	0.0	***3.5***	0.0	***27.1***

Table 2.4: Adaptation from synthetic to real. Judy Hoffman et al. [21] use SYNTHIA as source labelled training data and CityScapes train as an unlabelled target domain. Here GA represents global domain alignment, and CA indicates category-specific adaptation. Table is from [21]. Permission is requested.

SYNTHIA → CityScapes																				
Method	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIou
Dialation Frontend [71]	6.4	17.7	29.7	1.2	0.0	15.1	0.0	7.2	30.3	0.0	66.8	51.1	1.5	47.3	0.0	3.9	0.0	0.1	0.0	14.7
Their Method (GA only)	***11.5***	18.3	***33.3***	***6.1***	0.0	***23.1***	0.0	11.2	***43.6***	0.0	***70.5***	45.5	1.3	45.1	0.0	***4.6***	0.0	0.1	0.5	16.6
Their Method (GA + CA)	***11.5***	***19.6***	30.8	4.4	0.0	20.3	0.1	***11.7***	42.3	0.0	68.7	***51.2***	***3.8***	***54.0***	0.0	3.2	0.0	***0.2***	***0.6***	***17.0***

Table 2.5: Adaptation from synthetic to real. Yi-Hsuan Tsai et al. [56] used GTA5 as source labelled training data and CityScapes train as an unlabelled target domain. The table is from [56]. Permission is requested.

GTA5 → CityScapes																				
Method	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIou
FCNs in the Wild [21]	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
CDA [75]	74.9	22.0	71.7	6.0	11.9	8.4	16.3	11.1	75.7	13.3	66.5	38.0	9.3	55.2	18.8	18.9	0.0	16.8	14.6	28.9
CyCADA (feature) [20]	85.6	30.7	74.7	14.4	13.0	17.6	13.7	5.8	74.6	15.8	69.9	38.2	3.5	72.3	16.0	5.0	0.1	3.6	0.0	29.2
CyCADA (pixel) [20]	83.5	38.3	76.4	20.6	16.5	22.2	26.2	21.9	80.4	28.7	65.7	49.4	4.2	74.6	16.0	26.6	2.0	8.0	0.0	34.8
Theirs (singel-level) [56]	87.3	29.8	78.6	21.1	18.2	22.5	21.5	11.0	79.7	29.6	71.3	46.8	6.5	80.1	23.0	26.9	0.0	10.6	0.3	35.0
Baseline (ResNet)	75.8	16.8	77.2	12.5	21.0	25.5	30.1	20.1	81.3	24.6	70.3	53.8	26.4	49.9	17.2	25.9	6.5	25.3	36.0	36.6
AdaptSegNet (feature) [56]	83.7	27.6	75.5	20.3	19.9	27.4	28.3	27.4	79.0	28.4	70.1	55.1	20.2	72.9	22.5	35.7	8.3	20.6	23.0	39.3
AdaptSegNet (single-level) [56]	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
AdaptSegNet (multi-level) [56]	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

method is trained mainly by adversarial learning. Since the features in the segmentation task contain high-dimensional information such as shape and texture, they are quite complex and not easily transferable for learning. Yi-Hsuan Tsai et al. found that the two types of data are more visually consistent in the segmentation results by observing the characteristics of known and unknown data. Therefore, by performing DA on the output space of the network, training results can be obtained better than in the previously mentioned model. Figure 2.27 shows the similarity in this space.

Table 2.6: Adaptation from synthetic to real. Yi-Hsuan Tsai et al. [56] used SYNTHIA as source labelled training data and CityScapes train as an unlabelled target domain. The table is from [56]. Permission is requested.

SYNTHIA → CityScapes														
Method	road	sidewalk	building	light	sign	veg	sky	person	rider	car	bus	mbike	bike	mIou
FCNs in the Wild [21]	11.5	19.6	30.8	0.1	11.7	42.3	68.7	51.2	3.8	54.0	3.2	0.2	0.6	22.9
CDA [75]	65.2	26.1	74.9	3.7	3.0	76.1	70.6	47.1	8.2	43.2	20.7	0.7	13.1	34.8
Cross-City [8]	62.7	25.6	78.3	1.2	5.4	81.3	81.0	37.4	6.4	63.5	16.1	1.2	4.6	35.7
AdaptSegNet (single-level) [56]	78.9	29.2	75.5	0.1	4.8	72.6	76.7	43.4	8.8	71.1	16.0	3.6	8.4	37.6
Baseline (ResNet)	55.6	23.8	74.6	6.1	12.1	74.8	79.0	55.3	19.1	39.6	23.3	13.7	25.0	38.6
Theirs (feature)	62.4	21.9	76.3	11.7	11.4	75.3	80.9	53.7	18.5	59.7	13.7	20.6	24.0	40.8
Theirs (single-level)	79.2	37.2	78.8	9.9	10.5	78.2	80.5	53.5	19.6	67.0	29.5	21.6	31.3	45.9
Theirs (multi-level)	84.3	42.7	77.5	4.7	7.0	77.9	82.5	54.3	21.0	72.3	32.2	18.9	32.3	46.7

Figure 2.28 shows the network structure, which has two parts: the generator G and the discriminator D. Where generate G is composed of the segmentation network. The synthetic and real images are input to the segmentation network, and the softmax segmentation probability outputs P_s, P_t of the two images are obtained. These two feature maps are then input to the discriminator network D. D determine which domain the feature map comes from. The loss function of the model consists of two parts: segmentation loss and GAN loss. After tens of thousands of iterations, the model results are shown in Table 2.5 and Table 2.6. From the tables, we can see that the AdaptSegNet model has a much higher mIou than the FCNs, regardless of whether the GTA5 dataset is used as the source domain or the SYNTHIA dataset is used as the source domain.

Based on these theoretical foundations, we tried to improve the DA model and use it for semantic segmentation. Our final model is much more accurate than these models. We will focus on our model in the next chapter to show how it works.

Chapter 3

Proposed Model

In this chapter, we will focus on our unsupervised domain adaptation model, which firstly uses a fusion of source and target domain images to help the model perform initial domain adaptation. Secondly, with the help of GAN neural network, the model can better learn the similarity of the distribution between two domains and further perform domain migration. In this thesis, we name this model TwinLossGAN. The basic structure of this chapter is arranged as follows. In Section 3.1, we review the structure and basics of GAN neural networks. In Section 3.2, we will focus on the structure of our own model. In Section 3.3, we introduce the loss function used to train the whole model. In Section 3.4, we provide a comprehensive summary of what has been presented in this chapter.

3.1 Review of Traditional Unsupervised Domain Adaptation Model Based on Generative Adversarial Network

The traditional unsupervised domain adaptation model based on GAN is shown in Figure 3.1. In previous work, it is customary to directly pass the images of the source domain and target domain into two discriminators with shared weights. The semantic segmentation of the images in both domains is performed by two segmentors, and then the resulting segmentation results are fed into the discriminator, which determines which domain the segmentation results come from.

Due to the special structure of the generative adversarial network, the model can learn the distributional similarity between two different domains during the adversarial learning

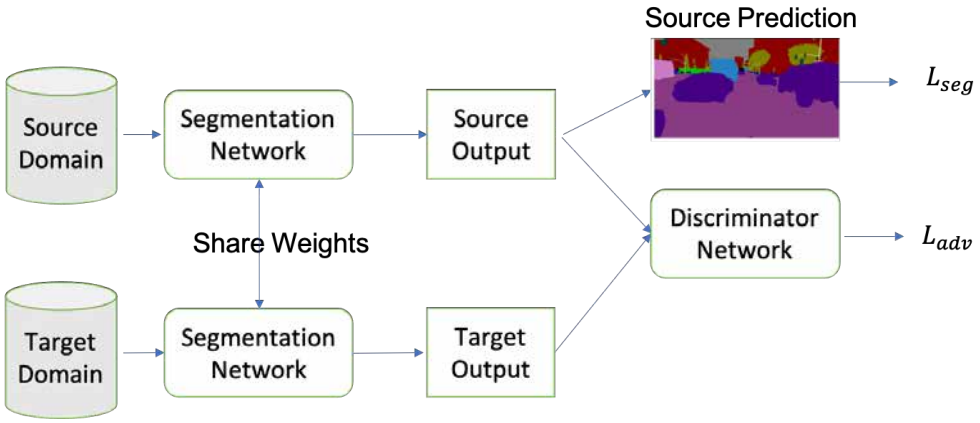

Figure 3.1: Previous model structure. The traditional domain adaptation model uses the source domain images and their ground truth to train the model. Then, the segmentation results of the model for different domain images are sent to the discriminator. The discriminator is used to determine the origin of the images.

process between the generator and the discriminator so that the image features learned from the source domain can be used to semantically segment the images in the target domain. GANs with different structures have been widely used in domain adaptation tasks in previous work. One of the classical models is [56]. This model mainly uses adversarial learning of generator and discriminator to close the distance of image distribution between two different domains, the source domain and target domain, in order to achieve the purpose of domain migration learning. However, this model also inherits the shortcomings of the GAN network. That is, the training process is unstable. We found in many experiments that the learning ability of a discriminator is significantly stronger than that of a generator. In order to train GAN stably, a lot of experiments are needed to tune the parameters. And because the source domain images are from synthetic data and the target domain images are from real data, the image distribution between the two domains is very different, so it is not very effective to use a GAN network to learn the difference between them. Therefore, we propose a new structure, as shown in Figure 3.2, and we will introduce our model structure in detail in the subsequent sections.

In summary, based on the problems of the previous model, the problems that need to be solved in our proposed model are:

1. The instability of GAN networks during training. This requires our model to be able to learn both adversarially and stably during the training of the generator and discriminator. Too much fluctuation does not help the GAN network to learn the image distribution

features.

2. The ability of GAN to reduce the inter-domain gap is limited. This requires our model to leverage other existing image processing methods to help the GAN network to further reduce the inter-domain gap. This can provide a good learning environment for GAN to apply the image features learned from the source domain to the target domain images.

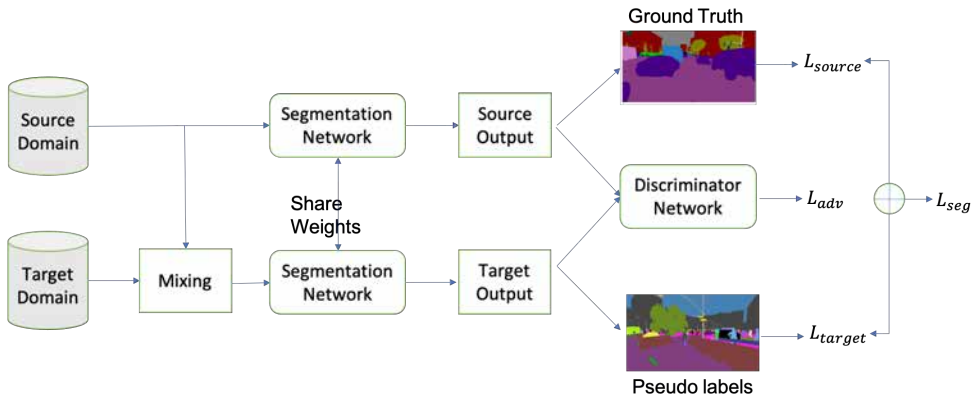

Figure 3.2: New model structure. In this kind of model, one more module called Mixing is added. This module can help the model narrow the gap between the source domain and target domain. Then, the model can generate a pseudo-label for target domain images. In this way, target images and their pseudo-label can help the model update its parameters. This kind of model can be well-trained.

3.2 TwinLossGAN

Motivated by the above model, we consider an unsupervised domain adaptation task for intensive prediction tasks. This model needs to address the two main problems of unstable GAN training and limited ability to reduce the gap between domains. In this setup, we assume that we have access to the source image set X_S, the source label set Y_S, and the unlabelled target image set X_T. Our goal is to train a model that can reliably and accurately predict the dense labels of each image in the target domain. An in-depth description of our model is given below.

Our domain adaptation model, shown in Figure 3.3, consists of three parts: the first part is the image fusion module M. The second part is the module G, which contains two

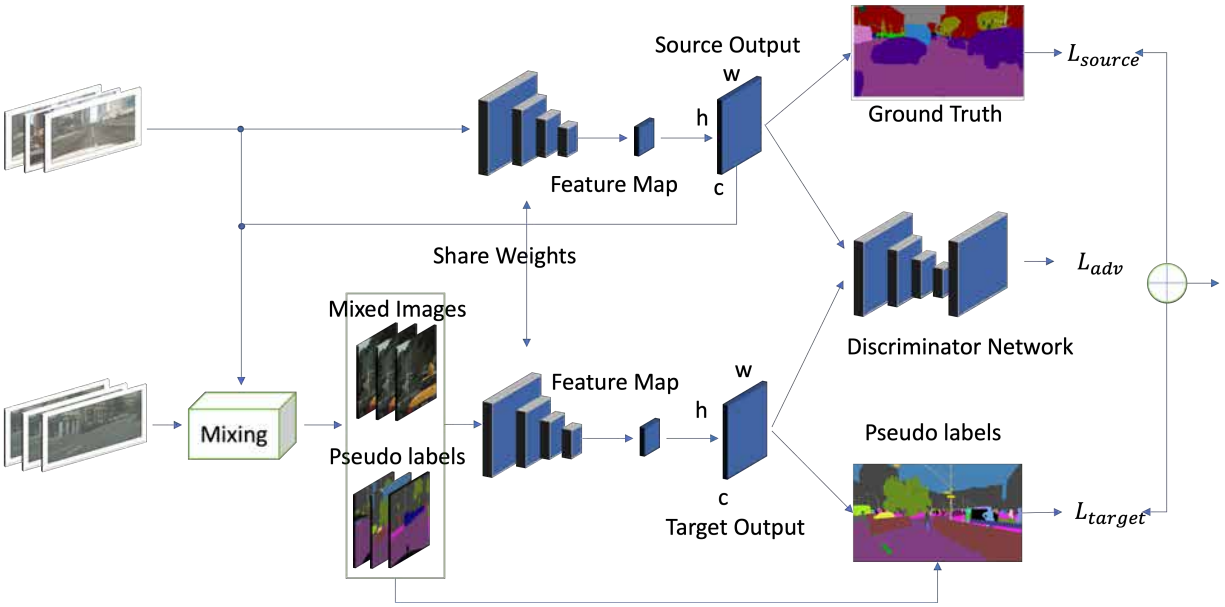

Figure 3.3: Structure of TwinLossGAN. our model is described clearly in section 3.2.

modules. One is the generator module G_S for source domain image segmentation, and the other is the generator module G_T for target domain image segmentation. The third part is the discriminator module D. The model inputs are images from two different domains. The images in the source domain are synthetic images. The labels of the synthetic images are visible to the model. The target domain image is the real image, and the label of the real image is not visible to the model. The goal of the G_S module is to semantically segment the images in the source domain under the supervision of the labels. Specifically, we input the image X_S of the source domain into the G_S module, which will first semantically segment the image and then compute the loss using the label set of the source image. The calculated loss is used to help the model to adjust the parameters. Finally, the source domain image can be segmented at the exact pixel level, and the segmentation result is noted as P_S. Then, the image in the target domain is processed, and before sending the image to the target domain image segmentor G_T, we will perform image fusion between different domains in the first step. A new synthetic image X_M and its corresponding pseudo-label Y_M are generated, and then X_M is fed into the generator module G_T, which is used for image segmentation in the target domain. Here the weights are shared between the two generator modules G_S and G_T. G_T fully learns the distribution features of the target domain images under the supervision of the pseudo-label Y_M. In addition, the target domain image is segmented at a pixel level. The segmentation results are recorded as P_T. The loss is computed using

the pseudo-label Y_M and P_T. G_T is optimized using backpropagation. This completes the task of the generator module. Then, we input the prediction results P_S and P_T from the two generator modules into the discriminator module D. The discriminator is used to determine whether the semantic segmentation results originate from the source domain or the target domain.

In our model, the generator also plays the role of segmentor. In the training process, in the same way as in the GAN network training, we first fix the discriminator to train the generator. The main role of the generator is to maximize the distribution of the images in the target domain as closely as possible to the distribution of the images in the source domain. In this way, the discriminator will not be able to easily distinguish the source of the final generated feature map. Then, we fix the generator and train the discriminator. The goal of the discriminator is that even if the generator brings the image distributions of two different domains infinitely close to each other, the discriminator can still distinguish whether the semantic segmentation results originate from the source or the target domain. In multiple adversarial learning, the model can fully learn the distribution features of source and target domain images to facilitate further domain migration.

The following describes how the model performs domain adaptation, which is divided into two stages. The first stage is a rough image fusion between domains before the images in the target domain are fed into the neural network. Specifically, two images, A and B, are sampled in the source and target domains, respectively. The prediction is performed separately using the segmentation model. The prediction results are generated by the argmax function in the prediction results of A. Then, half of the predicted labels are randomly selected as a mask, and the mask region in image A is merged into image B. The image fusion module generates synthetic images of source and target domains, and these synthetic images will have their own pseudo labels. In this stage, the distribution between the source and target images is brought closer together by the fusion, which helps the generator to learn the inter-domain gap more easily. This makes up for the problem we mentioned before: the discriminator learns too much. Our final goal is to make the segmentation prediction maps (P_S, P_M) of the source and target prediction images close to each other. So in the second stage, we input the segmentation prediction results of the two unused domains predicted by the generator to the discriminator to determine their sources. Under the adversarial loss of the target prediction, the network propagates the gradient from D to G, that is, G_S and G_T. This will encourage G to produce a similar segmentation distribution in the target domain as the source prediction. In order to make the model description clearer, we split the model and explain it step by step in the following sections.

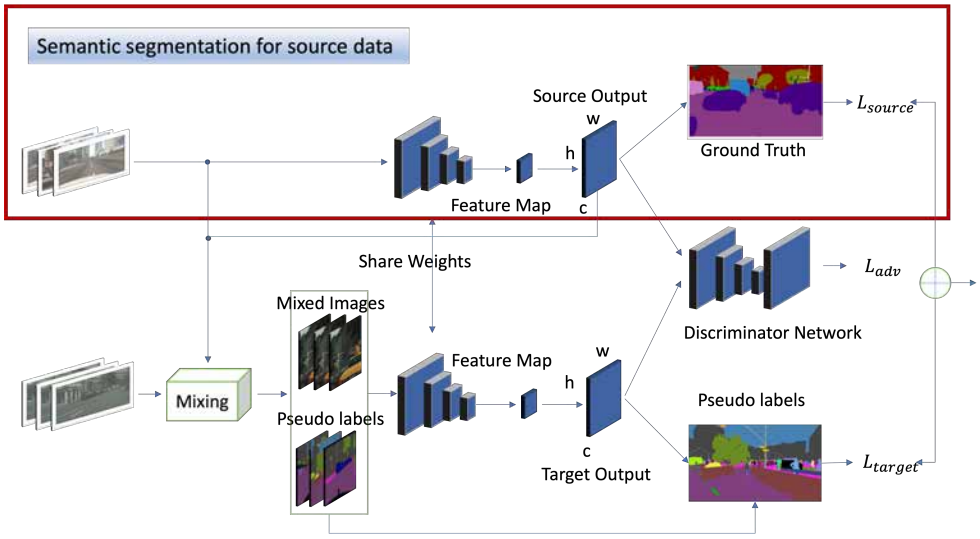

Figure 3.4: Semantic segmentation for source domain images.

3.2.1 Semantic Segmentation for Source Images

The red box in the Figure 3.4 indicates the process of training the model under the supervision of the source domain images. The images in the source domain and their labels are visible to the model. The images used in the source domain are synthetic images. The backbone is first given initial parameters. Then, we input the synthetic images into the backbone. The model performs semantic segmentation on the synthetic image. The obtained feature map is resized. Then, the obtained result is compared with the label. The calculated loss is used to update the parameters in the backbone. In summary, this is the training process for the source domain images for the model.

Figure 3.5: Images mixing result.

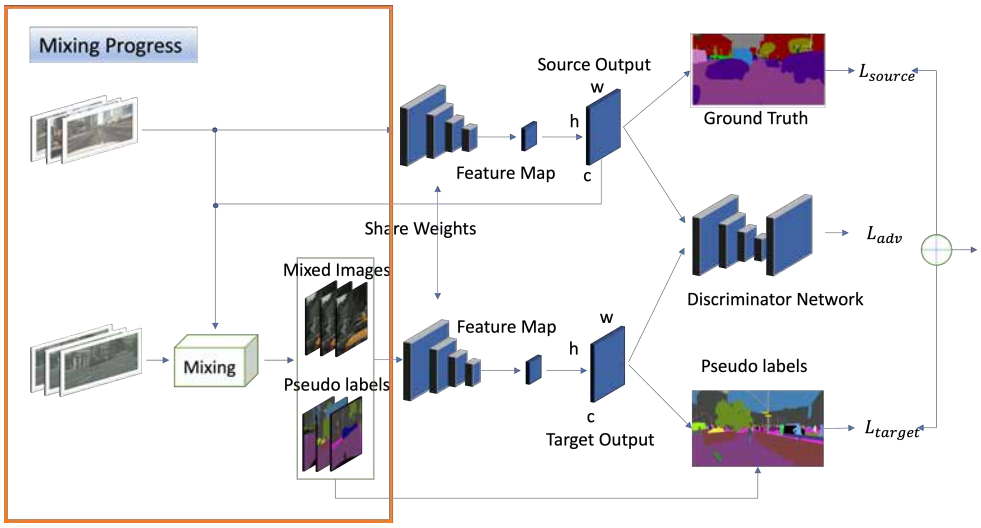

Figure 3.6: Images mixing result.

3.2.2 Mixing Progress

Before training with the target domain images, we utilize image enhancement methods. Image fusion of the source and target domains is performed first. The image fusion part is shown in the orange box in Figure 3.6. This is done by first copying the backbone structure mentioned in the previous section. Using the copied model, here labelled as a duplicate model, this model is used to semantically segment the images in the target domain. Then, we will have four input images. One of the images is from the source domain, and the other is the semantic segmentation result of the source domain image. Similarly, the other two images are the target domain images and the corresponding semantic segmentation results. We use the original images selected from the source and target domains for category fusion. By doing this, we select the categories in the source image to make a mask and use the mask to copy and paste the selected categories in the source domain to the target image, see Figure 3.5. This completes the blending of the images. In the same way, the semantic segmentation results of the two images can also be fused and used as labels for the new images. In summary, this concludes the image mixing part.

3.2.3 Semantic Segmentation for Target Images

The green box in the Figure 3.7 indicates that the model is trained using the images in the target domain. The images in the target domain are visible to the model, but the

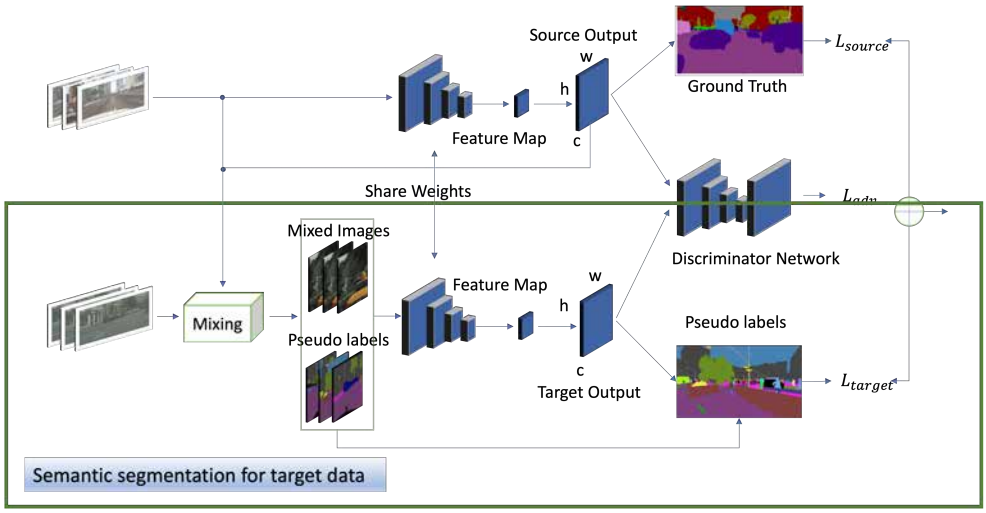

Figure 3.7: Semantic segmentation for target domain images.

labels of the images are not visible to the model. The model mentioned in the previous section recombines the source and target domain images to generate their corresponding labels. We feed the newly generated image into the model. The model performs semantic segmentation on this image. The labels are then used to compute the cross-entropy loss. The loss function will help the model update the parameters since the segmentors in the source, and target domains share parameters. So after parameter update, the model can learn the picture features of source and target domains better.

3.2.4 Discriminator

Discriminators play a crucial role in our entire model. The discriminator here (Figure 3.8) is similar to the discriminator in a GAN neural network. We feed both the semantic segmentation results obtained from the source domain image prediction and the semantic segmentation results obtained from the target domain image prediction into the discriminator. The discriminator needs to determine the source of the semantic segmentation results. Discriminating the source of the semantic segmentation results allows the two image distributions to approach each other through model learning. This facilitates the segmentor to apply the image distribution information learned from the source domain to the segmentation of the target domain images. In the overall model, the above two segmentors are equivalent to generators in GAN networks. Therefore, our model is trained based on the training mechanism of GAN neural networks.

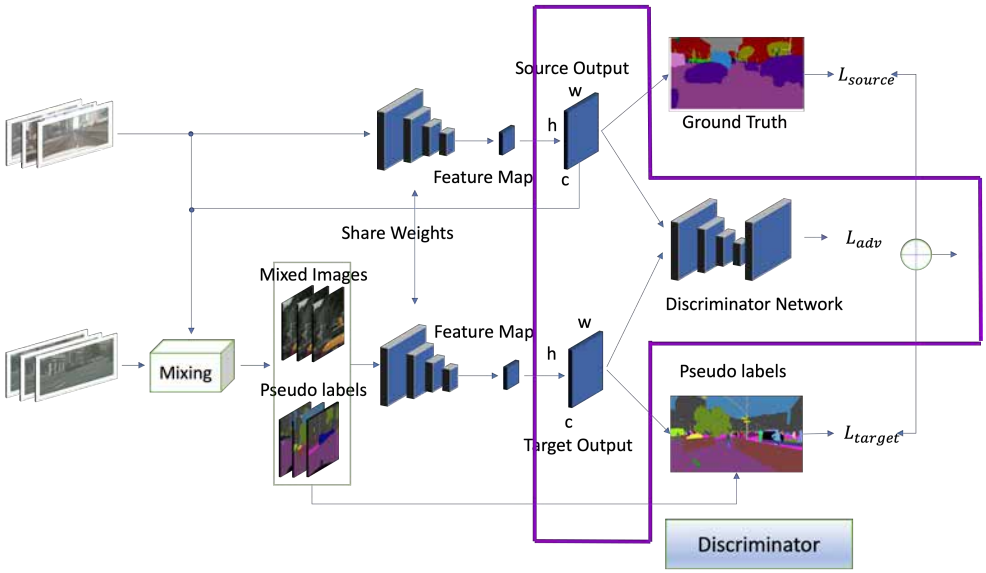

Figure 3.8: Discriminator in our model.

As you can see from the model description, the most important part of the model is the loss function. Our model contains three main loss functions, and we will focus on how these loss functions help train the model in the next section.

3.3 Loss Function

The loss function is crucial to the training of a model. It is used to evaluate the extent to which the predicted value of the model is different from the true value. A better loss function usually means a better model. Different models use different loss functions. There are two main types of loss functions. They are the empirical loss function and the structural risk loss function. The empirical loss function refers to the difference between the predicted and actual results. The structural risk loss function refers to the empirical risk loss function plus the canonical term. In this thesis, our model uses the empirical loss function in order to calculate the gap between the semantic segmentation domain and the ground truth performed by the model on the image. Thus, it helps the model to learn the image features better. Our model generates three loss functions, which we will introduce one by one below.

Algorithm 3.1 TwinLossGAN algorithm

Input: Source-domain and target-domain datasets D_S and D_T, segmentation network $\mathrm{f}_{\theta 1}$, discriminator $\mathrm{d}_{\theta 2}$, initialize network parameters θ_1 and θ_2 randomly

for $i = 1$ to n **do**

$X_S, Y_S \sim D_S$

$X_T \sim D_T$

$\tilde{Y_T} \leftarrow f_\theta(X_T)$

$X_M, Y_M \leftarrow$ *Generate new mixed images using images* X_S, Y_S, X_T *and* $\tilde{Y_T}$

$\tilde{Y_S} \leftarrow f_\theta(X_S)$

$\tilde{Y_M} \leftarrow f_\theta(X_M)$

$$For\ all\ \tilde{Y_S}, \tilde{Y_M} : d_{\theta 2}(\tilde{Y_S}, \tilde{Y_M}) = \begin{cases} 1, & \text{if the prediction is from source} \\ 0, & \text{if the prediction is from target} \end{cases}$$

$l \leftarrow l_S + l_M + l_d \triangleright$ *Losses from two segmentors and one discriminator,* L_S *and* L_M *are called twin losses*

Compute $D_\theta l$ *by backpropagation*

Perform SGD on θ

end for

return $f_{\theta 1}, f_{\theta 2}$

$$P_S = f(I_S) \tag{3.1a}$$

$$P_T = f(T_S) \tag{3.1b}$$

Equation 3.1a indicates that the images from the source domain are fed into the model to obtain the image prediction results. Where P_S denotes the prediction result of the model for the images from the source domain, it denotes the image from the source domain, and f denotes the model function used to perform semantic segmentation. Equation 3.1b indicates that the target domain images are fed into the model to get the prediction results. Where P_T denotes the prediction result of the model on the target domain image, I_T denotes the image is from the source domain and f is the same as Equation 3.1a.

$$I_M = M \odot I_S + (1 - M) \odot I_T \tag{3.2a}$$

$$G_M = M \odot P_S + (1 - M) \odot P_T \tag{3.2b}$$

To help the GAN-based neural network stabilizes its training process, the model performs semantic segmentation on the source and target domain images separately afterwards. We will use the source and target domain images and their prediction result maps to generate the mixed images. The formulas are shown in Equation 3.2a and Equation 3.2b. This is done by selecting a picture we and a picture T from the source and target domains, respectively. M is a binary mask that is used to randomly select half of the categories from the originating picture, set the pixels of these categories to 1 in M, and fill all other parts with 0. This mask is then used to generate the synthetic image I_M. By the same principle, using the prediction results of the model for images I and T, we can generate the ground truth G_M corresponding to the synthetic image I_M.

$$L_{source} = (I_S, G_S) = -\sum_{i=1}^{C} f_i(I_S) \log(G_S) \tag{3.3a}$$

$$L_{target} = (I_M, G_M) = -\sum_{i=1}^{C} f_i(I_M) \log(G_M) \tag{3.3b}$$

Equation 3.3a indicates that the model first predicts the source domain images and then computes the cross-entropy loss using its ground truth. Backpropagation with the computed loss can guide the whole model to learn the distribution features of the source domain images. Cross entropy is an important concept in information theory, which is mainly used to measure the similarity between the prediction results and the real results. Here we use this property to calculate the gap between the semantic segmentation results predicted by the model and the true labels. Equation 3.3b represents the loss generated by training the model with mixed images and their pseudo-labels. The principle is the same as in Equation 3.3a.

$$L_d = -\sum_{h,w} (1-z) \log\big((p)^{(h,w,0)}\big) + z \log\big((p)^{(h,w,1)}\big) \tag{3.4a}$$

$$L_{adv}(I_T) = -\sum_{h,w} (D(P_T)^{(h,w,1)}) \tag{3.4b}$$

Equation 3.4a and Equation 3.4b are used to represent the generative adversarial learning process. L_d denotes the loss generated by the discriminator, which is used to train the discriminator to accurately distinguish the prediction results from the source domain or

the target domain. L_{adv} denotes the loss generated by the generator, which is used to help train the generator to fool the discriminator so that the discriminator cannot distinguish the source of the semantic segmentation results. The model undergoes multiple generative adversarial learning iterations, and the model can extract the source and target domain image feature more accurately, which is beneficial for domain migration learning.

$$L_{seg} = L_{sourse} + L_{target} + \lambda L_{adv} \tag{3.5}$$

In summary, the final loss used for semantic segmentation is L_{seg}. In this Equation 3.5, L_{source} is used to calculate the cross-entropy loss between the predicted result for the source domain image and the real label. This cross-entropy loss can be used to help the neural network optimize the internal parameters so that the generator module G_S, which is used for source segmentation, can better learn the image distribution characteristics of the source domain to obtain more accurate segmentation results. L_{target} is used to calculate the cross-entropy loss between the image prediction in the target domain and the pseudo label. The purpose is the same as above. The cross-entropy loss is used to further adjust the internal parameters of the model. L_{adv} is the adversarial loss, which is used to adapt the predicted segmentation of the target image to the distribution of the source prediction, which is more useful for migration learning. λ_{adv} is used to balance the weights of the two losses. The total loss is back-propagated for the training of the whole model and parameter updating. The final model is able to perform more accurate semantic segmentation of real-world images.

3.4 Summary

In this chapter, we introduced the traditional generative adversarial learning network GAN. Based on the characteristics of GAN networks, we applied it to our own model, which is TwinLossGAN. Its adversarial learning facilitates our model to learn the similarity of image features between the source and target domains. We also use another image enhancement method, the mixing method, in our model. It helps the GAN-based model to perform image fusion first. This helps to stabilize the whole training process of the model and also facilitates domain migration learning. Finally, we introduce the loss functions used in the model. They are essential for the training of the model. In the next chapter, we describe the experimental parameters and the results in detail.

Chapter 4

Experimental Analysis

In this chapter, we will present the experimental parameter settings and some experimental results. In order to prove that our model is a significant improvement over the original model, we tested our model using the current popular UDA test datasets GTA5 and SYNTHIA. The results are compared with other previous models. For ease of understanding, we will describe the two datasets we used in detail later. In this chapter, Section 4.1 introduces the experimental details, such as the experimental parameters and settings. In 4.2, we introduce in general the two datasets we used. In 4.3, we describe in detail the test results of our model on the GTA5 dataset and another test dataset, SYNTHIA. In Section 4.4, we conclude this chapter.

4.1 Implementation Details

Our model is based on the [56] model, and the general structure of the model has been described in the previous chapter. Our model has three parts: the image fusion module M, the generator modules G_S and G_T for source domain image segmentation and target domain image segmentation, respectively, and the discriminator module D. Among them, the generator modules G_S and G_T use the ResNet101 network implemented in the DeepLab V2 framework as the backbone, which is used for semantic segmentation of the source and target domain images. This backbone network is already pre-trained on ImageNet [10] and MSCOCO [29]. Here we directly use the DeepLab trained model for prediction. Most of the parameters we borrowed from [56].

4.1.1 Parameter Setting

The details of the network training are as follows. First, we re-scaled the source image to 760×1280 and the target image to 512×1024. After that, we extracted a fixed size image of 512×512 for the whole network training. Several attempts have been made to find out that the 512×512 image size is more conducive to the training of generative adversarial networks.

For the initial learning rate, we set it to 2.5×10^{-4} and use the gradient descent algorithm with Nesterov acceleration. Accelerated Gradient Descent is an improved version of the Gradient Descent algorithm (GD), which was first proposed by Nesterov in 1983. It has been shown that the Accelerated Gradient Descent algorithm is the best method among all gradient-based algorithms. However, the original Accelerated Gradient Descent algorithm can only handle smooth convex optimization problems. So far, Accelerated Gradient Descent has been extended to a wider range of types of convex optimization problems. It was used here early so that the model could converge quickly. A polynomial decay with an exponent of 0.9 is used to improve the overall learning ability of the model.

In the implementation of the image fusion module M, we used ClassMix and added image colour dithering and random blurring to fully fuse the images while performing image enhancement. For the training, we formed a batch of two original images and two mixed images to train the network for 150,000 iterations. The experiments were implemented using a GTX 3090 GPU with 24GB of RAM. The detailed performance analysis report can be found in the appendix **??**.

4.1.2 Training Details

Our model mainly performs domain migration learning. First, for semantic segmentation of images in the source domain, we used $segmenter_1$, denoted as S_1. S_1 segments the images and then computes the loss function using the labels of the images themselves. The parameters of the S_1 itself are updated using the loss function. Secondly, for the target domain images, we used an auxiliary segmentor with the same structure as S_1. This segmentor is designed to perform the mixing of source domain target domain images. This is done by selecting one image from the source domain, and one from the target domain denoted as A and B. Some categories in A are selected using the image mask. The corresponding pixels in image B are removed using the same mask. Then, the categories selected are pasted from A into B. This completes the mixing of the images. The next step is to semantically segment image B using the auxiliary segmentor. We generated pseudo-labels using the

same method as above. The segmentation result of the B picture is combined with the segmentation result of the source domain picture A generated by S_1. The pseudo-label corresponding to the blended image is then obtained. At this point, we feed the blended image to $segmenter_2$, denoted as S_2. Here S_1 and S_2 have the same structure and share weights. Migration learning is performed between these two segmentors. Subsequently, S_2 performs semantic segmentation of the mixed image. Here the main segmentor is trained. It should be noted that here the segmentor acts as a generator in the GAN network, continuously learning the similarity of image distribution between two domains to fool the discriminator, and then we send all the semantic segmentation results generated by the segmentor to the discriminator to determine whether the results originate from the source or target domain.

4.2 Dataset

We show the experimental results of the UDA semantic segmentation model in this chapter. We used the synthetic dataset as the source domain to train the model. The trained model was then used to perform semantic segmentation on the real images, which are the target domain images. The source domain dataset used two commonly used synthetic image datasets, GTA5 [47] and SYNTHIA [48]. The target domain dataset used was Cityscapes. It was used to evaluate the learning results of the model on the source domain dataset. Before presenting the model results, we introduce the two datasets.

4.2.1 GTA5

As neural networks get larger and deeper, they are also more and more demanding on the dataset. The reliance on manually labelled images is no longer adequate for training large neural networks. In 2016 Stephan R. Richter et al. proposed that images of in-game scenes from modern games can be extracted and relevant labels made from them. GTA5 contains 25,000 synthetic images. The number of categories labelled in these synthetic images is 19, which is the same number of categories labelled in Cityscapes (shown in Figure 4.1).

Richter et al. [47] conducted experiments using their own generated dataset. The results (shown in Table 4.1) of the semantic segmentation experiments show that using the acquired data to fill in real-world pictures can significantly improve the accuracy of the model. The first experimental component is that the authors performed control experiments on the CamVid dataset. From the very beginning, when the full CamVid was used

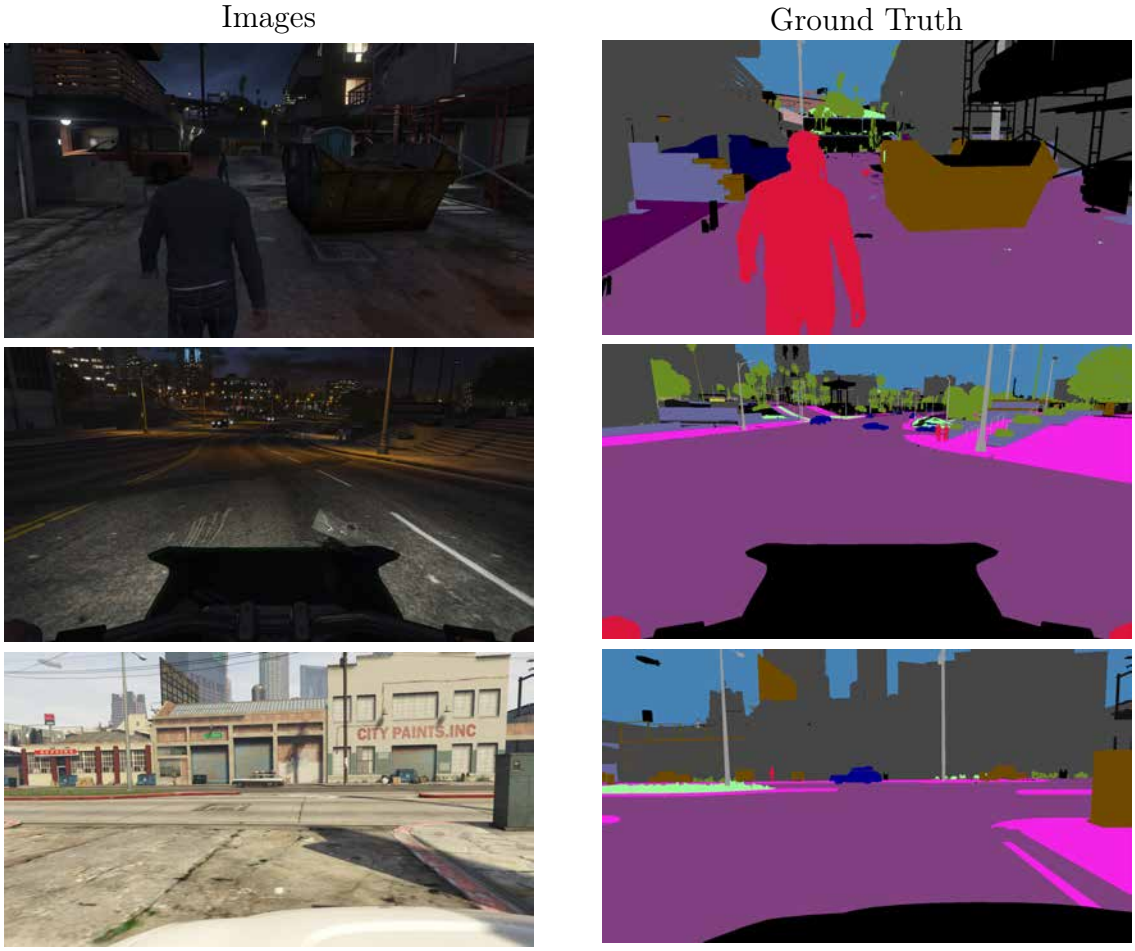

Figure 4.1: Images and labels in GTA5. Images are from [47]. Permission is requested.

for training, to later when all synthetic images were used, they found that the final accuracy could be improved by 3.9%.

4.2.2 SYNTHIA

SYNTHIA [48] is a composite dataset consisting of a set of realistic frames rendered from a virtual city, containing 9400 composite images (shown in Figure 4.2). SYNTHIA comes with precise pixel-level semantic annotations for 13 categories: miscellaneous, sky, buildings, roads, sidewalks, fences, vegetation, poles, cars, signs, pedestrians, bicyclists and lane

Table 4.1: Controlled experiments on the CamVid dataset. The table is from [47]. Permission is requested.

real images	100%	-	25%	33%	50%	100%
synthetic images (all)	-	100%	✓	✓	✓	✓
Intersection-over-union (Iou)	65.0	43.6	63.9	65.2	66.5	***68.9***

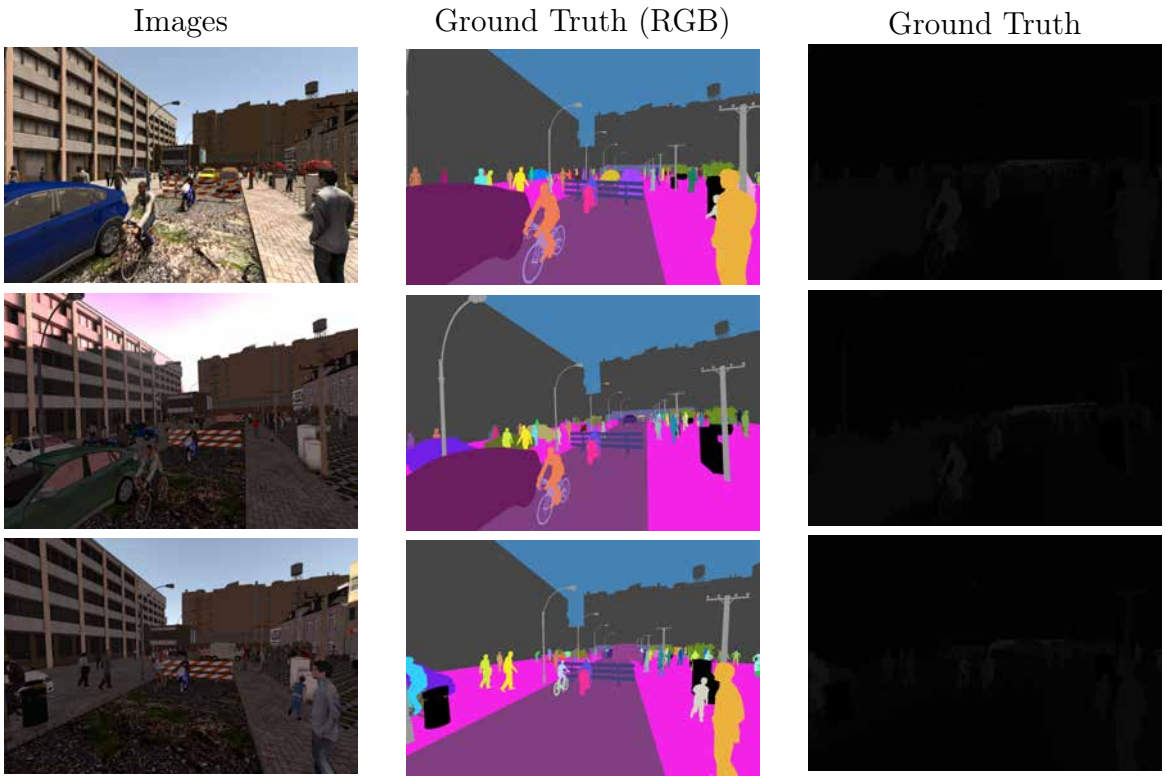

Figure 4.2: Images and labels in SYNTHIA. Images are from [48]. Permission is requested.

markings.

SYNTHIA is a very popular synthetic dataset at the moment. Scenes are taken from 200,000 video streams of HD images and 20,000 individual snapshots of HD images. It also contains a variety of scenes, such as European towns, highways, etc. Various dynamic objects are also marked: cars, bicyclists, etc. SYNTHIA also provides multi-seasonal images, as different lighting conditions and seasonal changes have an impact on the prediction

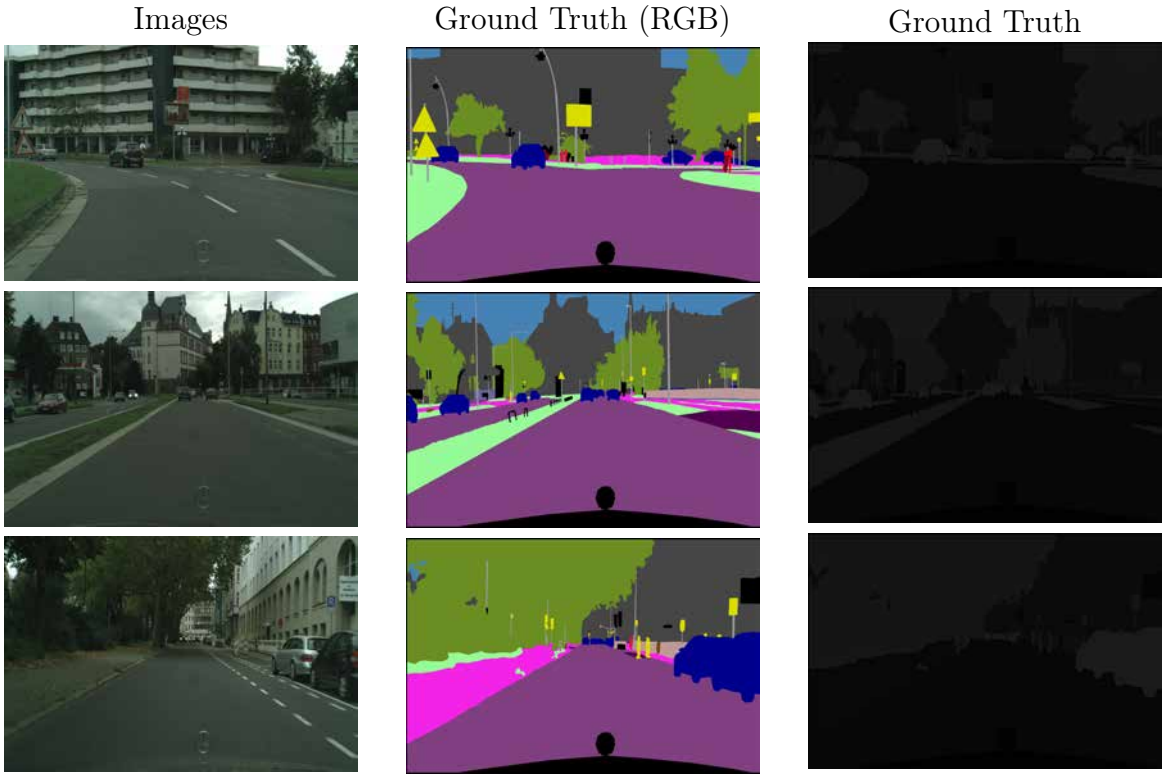

Figure 4.3: Images and labels in Cityscapes. Images are from [9]. Permission is requested.

results of the neural network model. In terms of lighting and weather, SYNTHIA offers, for example, a rain and night mode and a night mode. To sum up, SYNTHIA is an image information-rich synthetic image library. This is why we chose it as the original dataset.

4.2.3 Cityscapes

Cityscapes [9] is a dataset of semantic understanding images of urban street scenes. Cityscapes contains street scenes from 50 cities with different scenes, backgrounds and seasons, providing 5000 high-quality pixel-level annotated images of driving scenes in urban environments (2975 for training, 500 for validation, 1525 for testing, in total 19 categories; Samples shown in Figure 4.3). In addition to the 5000 frames of high-quality pixel-level annotations, Cityscapes has 20,000 coarsely annotated images (gt coarse). Therefore, the dataset is much larger in order of magnitude than the previous dataset.

The Cityscapes dataset aims to evaluate the performance of vision algorithms in two

major tasks of semantic urban scene understanding: pixel-level and instance-level semantic annotation. It supports research aimed at exploiting large amounts of (weakly) annotated data. The Cityscapes dataset, for example, is promoted by Mercedes-Benz and provides image segmentation datasets in automated driving environments. It is used to evaluate the performance of vision algorithms for semantic understanding of urban scenes. The performance of the algorithm is evaluated using the Iou score of the PASCAL VOC standard.

In this thesis, Cityscapes was chosen as the target domain. The semantic segmentation of real images within the dataset was performed using the trained model.

4.3 Results

In this section we will show the results of our model. We use the Iou as our criterion to evaluate the accuracy of the model. IoU measures the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks. Then we compute the Iou scores for each class separately and average them over all classes to provide the global average Iou which is mIoU scores for our semantic segmentation predictions.

Table 4.2: GTA5 to Cityscapes, our results are shown as per-class Iou and mIoU. We also compare our results to several previous works.

Method	road	sidewalk	building	wall	fence	pole	light	sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motocycle	bicycle	mIou
ResNet [18]	75.8	16.8	77.2	12.5	21	25.5	30.1	20.1	81.3	24.6	70.3	53.8	26.4	49.9	17.2	25.9	6.5	25.3	36	36.6
AdaptSegNet(s) [56]	86.5	25.9	79.8	22.1	20	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
DCAN [66]	85	30.8	81.3	25.8	21.2	22.2	25.4	26.6	83.4	36.7	76.2	58.9	24.9	80.7	29.5	42.9	2.5	26.9	11.6	41.7
DLOW [15]	87.1	33.5	80.5	24.5	13.2	29.8	29.5	26.6	82.6	26.7	81.8	55.9	25.3	78	33.5	38.7	0	22.9	34.5	42.3
AdaptSegNet(m) [56]	86.5	36	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.	35.4	3.9	0.1	28.1	42.4
SIBAN [33]	88.5	35.4	79.5	26.3	24.3	28.5	32.5	18.3	81.2	***40***	76.5	58.1	25.8	82.6	30.3	34.4	3.4	21.6	21.5	42.6
CLAN [50]	87	27.1	79.6	27.3	23.3	28.3	***35.5***	24.2	83.6	27.4	74.2	58.6	28	76.2	33.1	36.7	6.7	***31.9***	31.4	43.2
FDA [69]	90	40.5	79.4	25.3	26.7	30.6	31.9	29.3	79.4	28.8	76.5	56.4	27.5	81.7	27.7	45.1	***17***	23.8	29.6	44.6
FDA-ENT [69]	90.8	42.7	80.8	28.1	26.6	31.8	32.8	29.1	81.6	31.2	76.2	56.9	27.7	82.8	25.3	44.1	15.3	21.1	30.2	45
ABStruct [3]	91.5	47.5	82.5	31.3	25.6	33	33.7	25.8	82.7	28.8	82.7	***62.4***	***30.8***	85.2	27.7	34.5	6.4	25.2	24.4	45.4
AdvEnt [61]	89.4	33.1	81	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
MRNet [61]	89.1	23.9	82.2	19.5	20.1	33.5	42.2	39.1	85.3	33.7	76.4	60.2	33.7	86.0	36.1	43.3	5.9	22.8	30.8	45.5
CBST [83]	91.8	53.5	80.5	32.7	21	***34***	28.9	20.4	83.9	34.2	80.9	53.1	24	82.7	30.3	35.9	16	25.9	***42.8***	45.9
APODA [68]	85.6	32.8	79.0	29.5	25.5	26.8	34.6	19.9	83.7	40.6	77.9	59.2	28.3	84.6	34.6	49.2	8.0	32.6	39.6	45.9
PatchAlign [57]	92.3	51.9	82.1	29.2	25.1	24.5	33.8	33.0	82.4	32.8	82.2	58.6	27.2	84.3	33.4	46.3	2.2	29.5	32.3	46.5
FCAN [76]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.6
Source	63.31	15.65	59.39	8.56	15.17	18.31	26.94	15.00	80.46	15.25	72.97	51.04	17.67	59.68	28.19	33.07	3.53	23.21	16.73	32.85
Ours	***94.33***	***62.92***	***85.82***	***38.16***	***35.49***	31.66	35.32	***35.48***	***84.82***	34.65	***84.03***	59.52	17.48	***87.58***	***50.11***	***47.57***	0.02	12.37	0.3	***47.24***

4.3.1 GTA5 → Cityscapes

We show the results for GTA5 → Cityscapes in Table 4.2 and compare our models with previous models, all of which use the DeepLabV2 framework as the backbone. In the table, "Source" refers to a model that was trained only on the source data and then evaluated on the target data as the benchmark used for comparison in our table. From Table 4.2, we can see that our model performs very well on many objects compared to the previous model. For example, it has high accuracy on small objects such as signs and fences and large objects such as roads and sidewalks. As shown in Table 4.2, the final mIou accuracy of our model can reach 47.24%.

In addition to the table, we also provide a comparison with the original model (Figure 4.5), from which we can see that our model has a much higher segmentation accuracy than the previous model for sidewalk and road signs. Further, as can be seen in Figure 4.4, the road sign circled by the orange box is regarded as a small object in the semantic segmentation. The road sign is very fuzzy in the prediction result of the original model. In addition, the pixels assigned to the road sign classification are obviously insufficient. Additionally, in the same position in our model, it can be seen that the semantic segmentation of the road sign classification has been significantly improved compared with the previous model. Additionally, the blue box part can fully prove that our model has been improved. The blue area has two parts, lawn and ground. It can be seen from the semantic segmentation result of the original model. It cannot distinguish the lawn part from the ground part well. But in the results generated by our model, it can be seen that there is a clear separation between the lawn and the ground. This fully proves that our model is better than the original model for detecting large objects. These examples convincingly demonstrate that our model can fully learn the features of the synthetic image in the source domain and can be well applied to the image segmentation of the real image.

Table 4.3: SYNTHIA to Cityscapes, our results are shown as per-class Iou and mIoU. We also compare our results to several previous works.

Method	road	sidewalk	building	light	sign	veg	sky	person	rider	car	bus	mbike	bike	mIoU
Baseline(ResNet) [18]	55.6	23.8	74.6	6.1	12.1	74.8	79	55.3	19.1	39.6	23.3	13.7	25	38.6
AdaptSegNet(feature) [56]	62.4	21.9	76.3	11.7	11.4	75.3	80.9	53.7	18.5	59.7	13.7	20.6	24	40.8
IBAN [33]	78.2	19.7	***80.5***	9.4	8.9	77.4	82	56.3	9.6	76.3	22.8	17.5	23.3	43.2
AdaptSegNet(s) [56]	79.2	37.2	78.8	9.9	10.5	78.2	80.5	53.5	19.6	67	29.5	***21.6***	31.3	45.9
SIBAN [33]	82.5	24	79.4	***16.5***	12.7	79.2	82.8	***58.3***	18	79.3	25.3	17.6	25.9	46.3
PatchAlign [57]	82.4	38.0	78.6	3.9	11.1	75.5	***84.6***	53.5	21.6	71.4	32.6	19.3	31.7	46.5
AdaptSegNet(m) [56]	***84.3***	***42.7***	77.5	4.7	7	77.9	82.5	54.3	21	72.3	32.2	18.9	32.3	46.7
Source	36.30	14.64	68.78	5.59	9.05	68.96	79.38	52.45	11.34	49.77	9.53	11.03	20.66	33.65
Ours	75.52	14.96	80.19	3.55	***26.52***	***84.32***	73.68	54.05	***23.12***	***82.1***	***38.17***	13.97	***48.85***	***47.62***

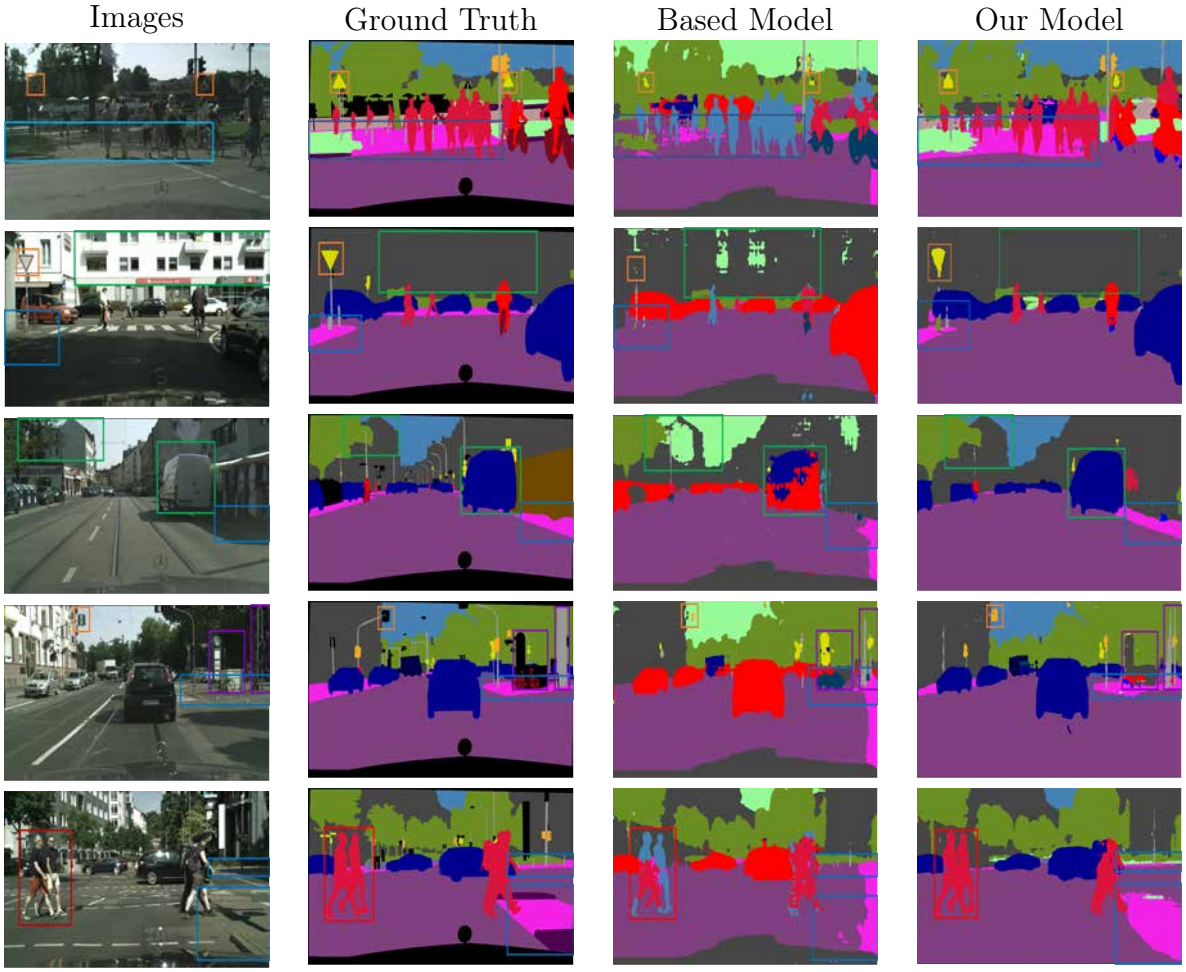

Figure 4.4: The figure shows a comparison between our model and the previous base model, showing that our model is more accurate in detecting smaller objects and larger objects.

4.3.2 SYNTHIA → Cityscapes

We show the results of SYNTHIA → Cityscapes in Table 4.3 for the SYNTHIA dataset, which contains only 16 of the 19 cityscape categories. In our experiments, we selected 13 categories to evaluate our model and compared our experimental results with the previous model. From the table, we can see that the models trained using the SYNTHIA dataset are not particularly outstanding compared to other models. However, although it is not as good as other models for some large objects such as roads, sidewalks, etc., it has high semantic segmentation accuracy for some small and medium-sized objects such as signs, riders, cars,

buses, etc. This further confirms our model's performance. This further confirms the outstanding performance of our model in the recognition task of smaller objects. The final mIou accuracy can reach 47.62%.

However, the model also has some shortcomings, the segmentation results for some large objects were not very satisfactory, and the model can be focused on the recognition of large objects in the subsequent research.

4.4 Summary

In this chapter, we gave a detailed description of the details of the model training and the equipment used. We then introduced all the datasets used for the experiments: GTA5, SYNTHIA and Cityscapes, and gave some sample images from the datasets. This provided an overall view of the datasets used for model training and testing. In this chapter, we also compared our model with other previous classical models, and we could clearly see the improvement of our model compared to other models in the table. In addition, we selected examples to illustrate the improvement of our model for small object prediction. At the end of this chapter, we also provided a graph of the model results and compare them with the base model and the ground truth.

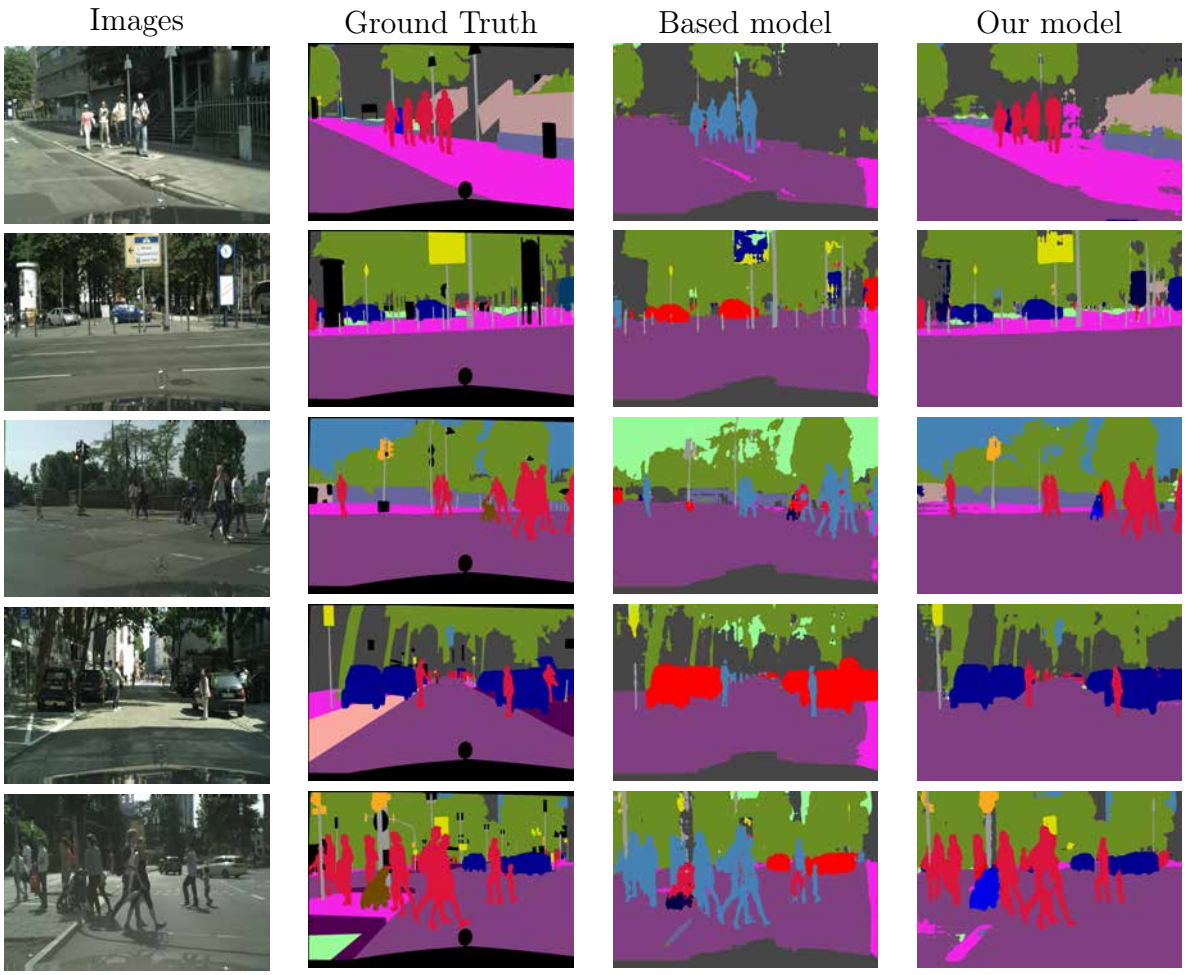

Figure 4.5: The model was trained on GTA5 and tested on Cityscapes. The original model cannot split the sidewalk and the road, but this problem does not occur very often in our model.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we proposed a new model, TwinLossGAN. This model can solve many problems found in traditional models. As mentioned in the previous sections, GAN networks have the problem of instability in the training process. The discriminator is much stronger than the generator. This will not allow the model to fully learn the distribution information of the synthetic images in the source domain. This results in the unsatisfactory segmentation results of the target domain images. Secondly, due to the limited learning ability of the GAN network combined with the UDA model, we introduced the inter-domain image fusion technique. This technique refers to the class-based fusion of the target domain images with the source domain images and generates pseudo-labels before feeding them into the model. This helps the model to better learn the similarity of the distribution between the source and target domain images and thus better apply the knowledge learned from the source domain to the target domain.

In this thesis, we trained our model using two synthetic datasets, GTA5 and SYNTHIA and tested it on a real dataset, Cityscapes, to show the advantages of our model in comparison to previous model results. When using GTA5 as the source domain dataset, our model had higher semantic segmentation accuracy than the traditional model for very large and very small objects. While using the SYNTHIA dataset as the source domain dataset, our model did not detect larger objects as well as the traditional model. However, the detection of small and medium-sized objects is better than the previous traditional model.

5.2 Future Work

Although our model improved in accuracy compared to the traditional model, many issues still need to be discussed. First of all, we encountered a problem where the segmentation accuracy of the model for larger objects was higher than that of the traditional model when the GTA5 training set was used as the source domain, but the semantic segmentation accuracy for larger objects was not as good as the conventional model when SYNTHIA was used as the source domain. One possibility is that SYNTHIA data and images are too complex compared with GTA5 objects, i.e., the overlap of objects is too high. After conducting image fusion in the source and target domains, the image complexity only increased further. It meant that our model was unable to learn how to distinguish between different objects smoothly, which led to a loss of accuracy.

Secondly, we found that the model could not meet the semantic segmentation accuracy requirements for objects of different sizes, which is also a point for future improvement. We hope that the model can deliver high segmentation accuracy on various sizes of objects. At present, we believe that if we want to continue to improve the accuracy of the model, it will be necessary to use other auxiliary information, such as depth information. Depth, as another type of essential image information, can help the model learn object contour information. In this way, we can theoretically improve the overall accuracy of the model.

Finally, the central part of our model uses a domain adaptation model. The biggest problem of the domain adaptation model is inter-domain variation. Too much difference between the distribution of images in two domains can seriously affect the learning of the domain adaptation model. However, on the other hand, we want to use synthetic images to train the domain adaptation model to perform semantic segmentation on real-world images. In this thesis, we used a GAN-based model combined with inter-domain image fusion to close the distance between two domains. The model was an improvement over the previous classical model. However, whether there are other ways to reduce the inter-domain gap faster is a question we need to consider further.

We will also continue to improve our model in the future. The improvement can be achieved in two ways, as summarized above. Firstly, we can add auxiliary information to enhance the learning ability of the model. Secondly, we can try to use a more advanced derivative version of the GAN network model as the main body, which may be of great help in model learning. More experiments will be conducted in the future to confirm our speculations.

References

[1] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *Advances in neural information processing systems*, 29, 2016.

[2] Chengtao Cai, Yue Wu, and Shuofeng Li. The application of the dilated convolution based on small object detection. In *2020 39th Chinese Control Conference (CCC)*, pages 7079–7083. IEEE, 2020.

[3] Wei-Lun Chang, Hui-Po Wang, Wen-Hsiao Peng, and Wei-Chen Chiu. All about structure: Adapting structural information across domains for boosting semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1900–1909, 2019.

[4] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7354–7362, 2019.

[5] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 627–636, 2019.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous

convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[8] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1992–2001, 2017.

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[11] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[12] Geoff French, Samuli Laine, Timo Aila, Michal Mackiewicz, and Graham Finlayson. Semi-supervised semantic segmentation needs strong, varied perturbations. *arXiv preprint arXiv:1906.01916*, 2019.

[13] Geoffrey French, Timo Aila, Samuli Laine, Michal Mackiewicz, and Graham Finlayson. Consistency regularization and cutmix for semi-supervised semantic segmentation. *arXiv preprint arXiv:1906.01916*, 2(4):5, 2019.

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[15] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2477–2486, 2019.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[17] Junjun He, Zhongying Deng, and Yu Qiao. Dynamic multi-scale filters for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3562–3572, 2019.

[18] 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.

[19] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[20] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proceedings of MLR International Conference on Machine Learning*, pages 1989–1998. PMLR, 2018.

[21] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[22] Wei-Chih Hung, Yi-Hsuan Tsai, Yan-Ting Liou, Yen-Yu Lin, and Ming-Hsuan Yang. Adversarial learning for semi-supervised semantic segmentation. *arXiv preprint arXiv:1802.07934*, 2018.

[23] Jake Elwes. Latent space. [Online; accessed 11-April-2022]. https://www.jakeelwes.com/project-latentSpace.html.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[25] Kushal Ranjan. Low and high level features. [Online; accessed 17-April-2022]. https://www.quora.com/What-is-the-difference-between-high-level-features-and-low-level-features.

[26] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

[27] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016.

[28] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[30] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1385, 2015.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[32] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*, 2016.

[33] Yawei Luo, Ping Liu, Tao Guan, Junqing Yu, and Yi Yang. Significance-aware information bottleneck for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6778–6787, 2019.

[34] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[35] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[36] Tom Mitchell and Machine Learning McGraw-Hill. Edition, 1997.

[37] Sudhanshu Mittal, Maxim Tatarchenko, and Thomas Brox. Semi-supervised semantic segmentation with high-and low-level consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[38] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Cambridge, MA: MIT press, 2018.

[39] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[40] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1369–1378, 2021.

[41] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333, 2019.

[42] Deepak Pathak, Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional multi-class multiple instance learning. *arXiv preprint arXiv:1412.7144*, 2014.

[43] PLOS ONE. Model of deeplab v2 structure. [Online; accessed 29-March-2022]. https://doi.org/10.1371/journal.pone.0246093.g004.

[44] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[45] Receptive Field Size in CNN. Receptive field. [Online; accessed 17-April-2022]. https://www.baeldung.com/cs/cnn-receptive-field-size.

[46] Renu Khandelwal. Teacher network and student network. [Online; accessed 21-April-2022]. https://medium.com/analytics-vidhya/knowledge-distillation-in-a-deep-neural-network-c9dd59aff89b.

[47] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[48] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[49] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, and Elisa Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9471–9480, 2019.

[50] Congcong Ruan, Wei Wang, Haifeng Hu, and Dihu Chen. Category-level adversaries for semantic domain adaptation. *IEEE Access*, 7:83198–83208, 2019.

[51] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[52] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997. PMLR, 2017.

[53] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[54] Kihyuk Sohn, Wenling Shang, Xiang Yu, and Manmohan Chandraker. Unsupervised domain adaptation for distance metric learning. In *International Conference on Learning Representations*, 2018.

[55] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1379–1389, 2021.

[56] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7472–7481, 2018.

[57] Yi-Hsuan Tsai, Kihyuk Sohn, Samuel Schulter, and Manmohan Chandraker. Domain adaptation for structured output via discriminative patch representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1456–1465, 2019.

[58] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[59] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[60] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[61] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2517–2526, 2019.

[62] Hua Wang, Cuiqin Ma, and Lijuan Zhou. A brief review of machine learning and its application. In *2009 international conference on information engineering and computer science*, pages 1–4. IEEE, 2009.

[63] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[64] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. Ieee, 2018.

[65] Wikipedia. Random field. [Online; accessed 12-April-2022]. https://en.wikipedia.org/wiki/Random_field.

[66] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S Davis. Dcan: Dual channel-wise alignment networks for unsupervised scene adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 518–534, 2018.

[67] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International conference on machine learning*, pages 5423–5432. PMLR, 2018.

[68] Jihan Yang, Ruijia Xu, Ruiyu Li, Xiaojuan Qi, Xiaoyong Shen, Guanbin Li, and Liang Lin. An adversarial perturbation oriented domain adaptation approach for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12613–12620, 2020.

[69] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4085–4095, 2020.

[70] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[71] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[72] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[73] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[74] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3801–3809, 2018.

[75] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2020–2030, 2017.

[76] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Fully convolutional adaptation networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6810–6818, 2018.

[77] Youshan Zhang. A survey of unsupervised domain adaptation for visual recognition. *arXiv preprint arXiv:2112.06745*, 2021.

[78] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.

[79] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[80] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[81] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[82] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Domain adaptation for semantic segmentation via class-balanced self-training. *arXiv preprint arXiv:1810.07911*, 2018.

[83] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.