VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

NGUYEN THANH DANH – PHAN NGUYEN

THESIS

SEMANTIC IMAGE SEGMENTATION IN THE DARK WITH DOMAIN ADAPTATION METHOD

HONORS BACHELOR IN COMPUTER SCIENCE

HO CHI MINH CITY, 2021

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

NGUYEN THANH DANH – 17520324
PHAN NGUYEN – 17520828

THESIS

SEMANTIC IMAGE SEGMENTATION IN THE DARK WITH DOMAIN ADAPTATION METHOD

HONORS BACHELOR IN COMPUTER SCIENCE

THESIS ADVISOR
NGUYEN VINH TIEP, Ph.D.

HO CHI MINH CITY, 2021

ASSESSMENT COMMITTEE

The Assessment Committee is established under the Decision. ...., date .... by Rector of the University of Information Technology.

1. Assoc. Prof. Ly Quoc Ngoc – Chairman.
2. M.Sc. Nguyen Thanh Son – Secretary.
3. Ph.D. Nguyen Hong Phuc – Member.

Acknowledgement

It is not possible to prepare a dissertation without the assistance and encouragement of other people. This one is certainly no exeption. Thus, this dissertation is dedicated to those who make it possible:

First of all, we would like the express our deep gratitude towards our advisor PhD. Nguyen Vinh Tiep, who inspired us the passion of research, for his instructions and supports during the process of our work.

Besides, we would like to thank the Faculty of Computer Science - University of Information Technology where we have been prepared enough knowledge to pursuing the field of Computer Science research. We also thank Multi Media Laboratory (MMLab-UIT) for supporting us the research environment and cutting-edge devices throughout the university journey.

Specially, we want to say thank you to our fellow friends who support us with precious ideas and are always willing to help us when we are in need.

Last but not least, we genuinely thank our families without whose love and support we would not have been here to complete this dissertation at the university.

ABSTRACT

Semantic image segmentation is a fundamental computer vision task that brings many conveniences to human life. Its applications vary in many aspects such as autonomous vehicles, medical systems, photo editors and so on. For instance in the field of autonomous vehicles, the vehicles take images from cameras along with other sensors to understand the surroundings. The taken images are automatically handled by computer vision algorithms including semantic image segmentation. In the normal condition of daytime, many previous work were finished with yielding high performance. However, in nighttime condition which we define as in the dark, there are several work done with poor results. To be specific, we discover that the specifications of images in daytime is different from those in nighttime mainly due to light condition. Furthermore, there is a limitation of annotated nighttime images for semantic image segmentation purpose. Focusing on the application of autonomous vehicles, we do the task of semantic image segmentation on nighttime cityscapes images. In this work, we propose a complete framework to solve the problem of semantic image segmentation in the dark with the help of domain adaptation method. To address the problem of lacking suitable annotated dataset, we leverage Generative Adversarial Network - GAN to transfer images from daytime to nighttime domain. In segmentation task, we apply self-training method and propose a loss function to enhance the performance of the model. Finally, we show the improvement and efficiency of our hypothesis in a series of experiments and lessons for future researches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

1.1.1 Practical Context

Along with the development of science and technology, the living standard of human being is significantly boosted. The conveniences have upgraded our life to the stage of living to enjoy the surroundings. In fact, work that used to be done by human are replaced with the help of robots. For instance, the household chores like floor sweeping, dish washing or meal cooking are undertaken by automatic machine more and more today. In traffic monitoring, instead of manually handling and navigating traffic flow, we have traffic cameras together with artificial intelligence system or computer vision system to analyse and support human to manage the traffic. Also with the help of technology, we stay closed to each other via social network and the internet despite physical distance. Technology is applied to every aspect of human life with the final target to make it more convenient and better performance.

The development of technology leads to the growth of computer science, in which computer vision plays an important role. It would be a deficiency if computer vision and its applications were not mentioned in the process of making our life more convenient. Let us take some exemplary applications to represent the robust potential of this field.

Autonomous vehicles. Means of transportation are invented to reduce the human effort when travelling among destinations. However, with an aim at tackling most of the task of drivers, autonomous vehicles are created. Autonomous vehicles receive information like images and signal from cameras, sensors and radars around the vehicle to compute for the next suitable move. Figure 1.1 illustrates the virtual vision of an autonomous vehicle. Those information are used to illustrate the surroundings of the vehicle for the computer to know and give suitable respond. In this task, computer vision handles images with some image processing methods to assist the computer in understanding the image [17, 1, 12, 34]. Overall, every single operation contributes to the complete work of the autonomous vehicle.

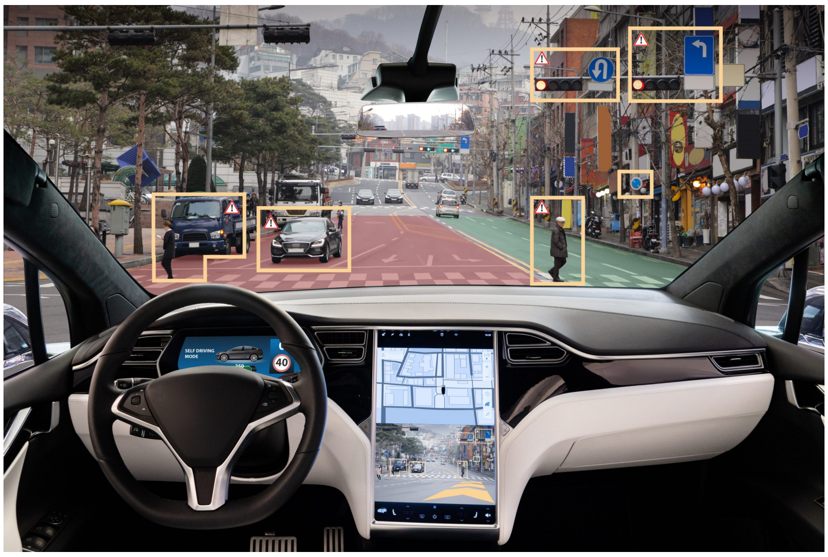

Figure 1.1: Illustration of the virtual vision of an autonomous car. Autonomous vehicles understand the surroundings with the help of cameras, sensors or radars.[1]

Medical recommendation system. From past to present, medical field is known to be related to taking care of human health. The task is usually done by experts or doctors who have specialized knowledge of medical. As to the fact that medical care is an obvious demand of every single person in the world, which leads to the enormous workload in this field. To be honest, machine cannot replace the position of experts in diagnosing symptoms of patients. However, it can help to promote recommendation which is helpful as reliable references for experts. To be specific for medical images (X-ray or MRI images), computer vision can help recognize abnormal signal with high accuracy [47, 24, 29, 42]. Experts who based on this prediction is able to shorten the process of medical examination as well as reduce work stress. Figure 1.2 shows a typical example of how accurate brain tumor segmentation help experts in their work.

[1]https://www.ptolemus.com/topics/autonomous-vehicles/

[2]https://news.developer.nvidia.com/automatically-segmenting-brain-tumors-with-ai/

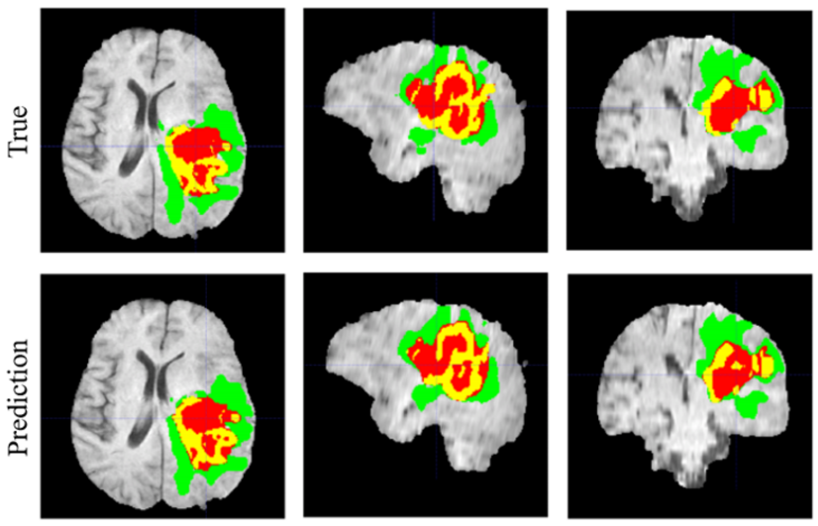

Figure 1.2: Artificial intelligence can well help automatically segment brain tumors.[2]

World map drawing/updating - Satellite image understanding. Map is a very functional tool for people who travel around without knowing the features destination place. When travelling, we leverage maps to find out which way to go or which direction to follow. In the past, map drawing is a really intense task that requires lots of manual work, from measuring the field to mapping the reality to the paper. Nowadays, with the helpful hand of technology, the image of the earth surface can be easily capture from satellite to convert to a map. Problem occurs when there are plenty of images with super large size. Moreover, maps require up-to-date accuracy while the world facilities change rapidly, which means maps need to be re-draw continually. Here, we need the intervention of technology to automate the process. Computer vision is able to to tackle this problem with automatically label objects in satellite images, which help the process of drawing a map faster [9, 10, 50, 23]. Figure 1.3 gives an example of building segmentation on satellite images, which can help automatically draw a map.

The three mentioned highlights of application above can be done with the approach of computer vision. To be specific, that approach is called **Semantic Image Segmentation**. Beside the highlights, there are plenty of applications of semantic image segmentation that have been appeared around us. Furthermore, this approach of semantic image segmentation is the bridging stage of many other problems.

[3]https://deepsense.ai/satellite-images-semantic-segmentation-with-deep-learning/

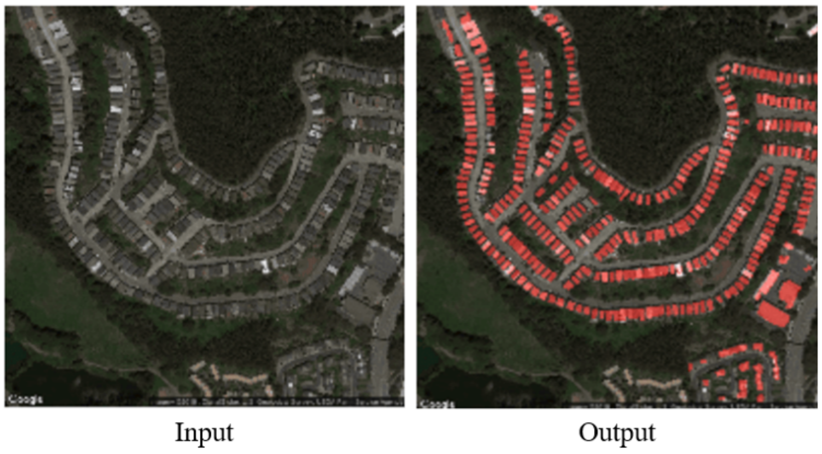

Figure 1.3: Satellite image segmentation helps automatically draw a map from an image.[3]

1.1.2 Problem Definition

Consider the importance of autonomous vehicles operation, we focus on applying semantic image segmentation to handle images taken from the view of the vehicles. The task of segmenting images help the computer to understand the surroundings of the vehicles in pixel level. Note that from the view of vehicle, the received images are cityscapes which includes traffic vehicles, traffic light, pedestrians and so on. Therefore, the problem we concerned is cityscapes image segmentation.

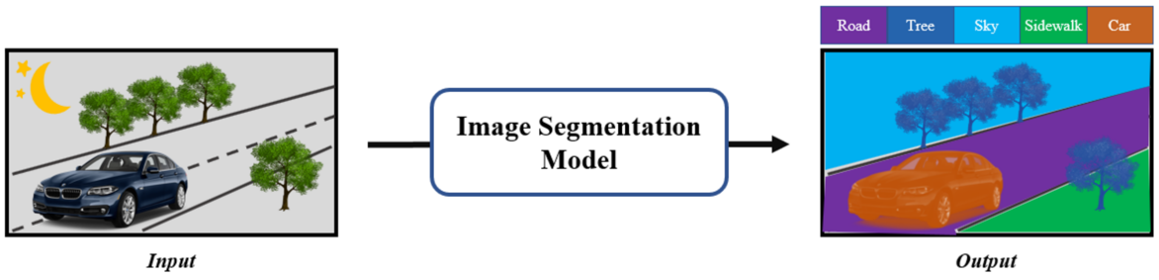

Figure 1.4: Definition of nighttime cityscapes image segmentation as a toy example. Image segmentation model take a nighttime cityscapes image as input and process to output a segmentation map that assigns each pixel a semantic label.

We surveyed that there were many work done on this topic but specific for **daytime images** only [44, 33, 28, 2, 3, 4, 5, 53]. On the contrary, similar work on nighttime images is limited. In fact, driving at night is not so rare, and there should be available Advanced Driver Assistance System (ADAS) that can well adapt both nighttime and daytime conditions. Therefore, we choose to perform **semantic image segmentation**

on nighttime cityscapes images in this thesis. The problem is described in Figure 1.4 with input is nighttime cityscapes images and output is the segmentation maps indicating which pixel belongs to which semantic objects.

1.1.3 Challenges

The most considerable challenge of our work is related to dataset. We are facing the problem of **lacking of training annotated nighttime cityscapes images** for the task of semantic image segmentation. Despite the fact that there exists many datasets of cityscapes, their specifications are in daytime.

Besides, as well as other computer vision tasks, semantic image segmentation on nighttime cityscapes images works with image data. Thus, there are many challenges of image data type:

- Image data is **enormous, diverse and complicated**.
- Image taken in different **negative conditions** such as low light, lightburn, backlight and so on or image with low quality due to low resolution or out of focus.
- Objects in the images suffer from **occlusion**.

1.2 Objectives

In this thesis, we focus on solving the biggest mentioned challenge of lacking of annotated nighttime cityscapes images by proposing a general framework. Meanwhile, we observed that there are daytime annotated cityscapes images datasets for semantic image segmentation task. It is obvious that the domains of these datasets and the ones we need are different. We come up with the idea of leveraging those daytime data to train a model that has the ability of handling semantic segmentation in nighttime condition. Therefore, our objectives is to propose a framework to tackle the problem of **semantic image segmentation in the dark** with the assistance of **domain adaptation method** along with making use of existing daytime annotated images. To be specific, we target at applying **GAN-based methods** to do the task of domain adaptation. In order to improve the segmentation performance, we make use of **self-training technique**. Altogether, we aim at optimizing the performance of our complete proposed framework.

1.3 Contributions

Our main contributions in this work are listed in three points:

- Propose a complete framework with domain adaptation method for semantic image segmentation in the dark along with making use of GAN-based methods and self-training technique.
- Propose a loss function that optimizes the semantic image segmentation model performance.
- Build a nighttime cityscapes dataset by GAN-based method for the task of semantic segmentation.

The proposed framework has two main components whose brief idea are mentioned below:

GAN-based Image Domain Translation. Generative Adversarial Network [11] (GAN) was widely used to deal with multiple problems, especially in image-to-image translation. Particularly, GAN simultaneously contains two models: a generative model G that catches the crucial distribution and a discriminative model D that evaluates the probability of sample from training data generated by G. Inspired by this idea, we use method based on GAN to convert cityscapes images from daytime to nighttime domain, which works as the first stage in our proposed method. Besides, nighttime images are also translated back to daytime domain to for the purpose of evaluating our trained model in ordinary condition.

Semantic Image Segmentation with Self-training. In this image segmentation component, we implement a segmentation model which takes the domain-translated images from the previous module as input. The model is trained to do the nighttime cityscapes image segmentation with the combined data of both daytime and nighttime images. Moreover, self-training technique is leveraged in this stage to improve the segmentation results. To be more specific, we choose a subset of true nighttime images of cityscapes to be our extra unlabeled data. The trained segmentation model take the responsibility of inferring pseudo-labels for those extra data. Then, altogether the labeled and unlabeled data are used to refine the trained model, this is the so-called self-training technique. Besides, in this component, we propose to combine the two common loss functions of cross entropy loss and focal loss to be the total loss of our segmentation model, which leads to better segmentation performance.

1.4 Dissertation Structure

Our thesis consists of 5 chapters:

- Chapter 1: Introduction. This chapter presents an overview of our work as well as the objectives and main contributions of our thesis.
- Chapter 2: Related Work. This chapter presents fundamental knowledge and researches that are related to our thesis.
- Chapter 3: Methodology. This chapter presents in details our proposed framework for semantic image segmentation on nighttime images with domain adaptation method and our modifications.
- Chapter 4: Experiments. This chapter reports our experimental results and the process of how we improved the performance of our framework.
- Chapter 5: Conclusion. This chapter summarizes our thesis and our main contributions, we also mention some future work.

Chapter 2

Related Work

2.1 Overview

This section briefly summarizes the whole information of the related work that is relevant to our work. Overall, there are four main points including Convolutional Neural Network, Semantic Image Segmentation, Generative Adversarial Network (GAN) [11] and Image Domain Adaptation. Firstly, we shortly introduce fundamental knowledge in CNN to recall the background. The first one shows the basic knowledge of CNN, its motivation and its components. We also analyse the impact of each component in a whole network. In the next Semantic Image Segmentation, we structure the historical work and outline the improvement path of semantic image segmentation. To be more detailed, we firstly point out the definition of semantic segmentation, which is followed by some classical networks involving Fully Convolutional Networks [31], Deconvolutional Network [35], U-net [39], Pyrmid Scene Parsing Network [53], RefineNet [25] and Google DeepLab Family [2, 3, 4, 5]. The next striking section mentions Generative Adversarial Network - a recent well-known method. After introducing GAN, a simple background of GAN is described such as an overview of GAN, its architecture, loss function and training method. The most attractive one is applications of GAN. Similar to segmentation, GAN also has its improved versions, namely Conditional GAN (CGAN) [32], Pix2Pix [16], CycleGAN [54]. Finally, Image Domain Adaptation is discussed in terms of low-light condition semantic segmentation with several previous work.

2.2 Fundamental Knowledge in Convolutional Neural Network

For the time being, Deep Learning is a brand of Machine Learning based on artificial neural via learning how to represent features. Generally, it can be learnt by supervised, unsupervised and semi-supervised methods. Deep Learning now such as Deep Neural Networks, Convolutional Neural Networks (CNN) and so on are applied to various fields like Computer Vision, Natural Language Processing, Voice Recognition, Machine Translation, Medical Image Analysis and other applications, which efficiently support the experts.

Artificial Neural Networks (ANN) has solved simple classification problems. However, some matters require complex and a large number of data, which leads to not able to use ANN due to the large number of parameters and expensive hardware. It is clearly explained that the fully connected layers with each node between consecutive layers results in vast parameters. In addition, two nodes seem to have spatial relation when they are located close to each other. Hence, to reduce the unnecessary parameters represented the spatial relationship information, local connection is applied instead of global connection. This is the motivation of CNN, which significantly decreases a large number of unnecessary parameters. Simply, local connection and weight-sharing are based on convolution operation. There is a tool - Deep Visualization Toolbox [51] that visualizes the effect of each layer in the CNN, which help to imagine the meaning of the layer. In theory, take a simple CNN for instance, a CNN contains several main layers:

Convolution Layer. Generally, the standard neural network has a large number of parameters because of fully connected layers. To diminish unnecessary parameters, each neural only links to a local part instead of the entire image. Convolution layers assumes that parameters of filters is the same at every position on the image. Another name of these parameters is kernel. Convolution is combined by various kernels. When Convolution is applied, output image size is preserved. We can decrease sampling rate by changing the stride. We also add padding by a prior value (usually is 0) which covers the border of the input image to preserve its size.

Non Linear (Activation) Layer. Basically, Convolution is a linear operation. If all neural are comprised by linear operation, a neural network will be a linear function. Then, this neural network is a logistic regression. To avoid this, each neural needs to be followed by a nonlinear activation function. There are several nonlinear activation, especially, recent researches prove that using ReLU activation function would get exponential results as the following aspects: (i) simple calculation, (ii) faster training and (iii) sparse property in hidden neural. For example, after initial weight, around 50% hidden neural are activated (values is greater than 0).

Pooling Layer. Pooling plays a vital role in CNN, which helps to select essential values on each channel to summarize the signal on each local part. It is widely used such as Max-Pooling, Sum-Pooling and L_2-Pooling. However, the most popular one is Max-Pooling which determines the strongest signal when applying the same filter on an image. Filter size depends on the variation of transformation, the larger size the greater deformation and vice versa.

Normalization Layer. Normalization function helps CNN to normalize the value of each pixel which depends on each channel. Some advantages of the normalization are: (i) invariance (e.g. illumination), (ii) improving the optimization of the training process and (iii) increasing sparse of neural. For example, the values of normalized pixels are fed into activation ReLU, which leads to the a wide range of values that equal to 0.

Fully Connected Layer. The purpose of this layer is to classify the extracted features from Convolution Layer. Fully Connected Layer comprises fully connected functions, which links to each other and ends up with activation function to represent the probability of classes.

2.3 Semantic Image Segmentation

2.3.1 Overview of Semantic Image Segmentation

Computer vision is an important sub-field of Artificial Intelligence with an aim at guiding the computer to do visual tasks in the same way as human eyes can do and understand images. Considered tasks in computer vision can be listed as image classification, object detection and localization or image segmentation. In this section, we focus on introducing image segmentation issue.

To our knowledge, image classification outputs the information of what object in the input, object localization and detection give one more information of the position of the object. Meanwhile, image segmentation has an ability of identifying exactly which pixels in the input image belong to the same semantic object. In other words, image

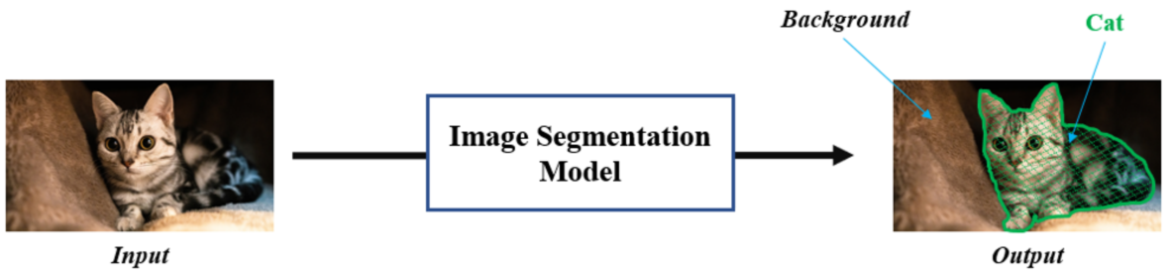

Figure 2.1: Illustration of semantic segmentation problem. The Image Segmentation Model identifies that there exists a cat in the input image and which pixels belong to the cat.

segmentation is the process of assigning the semantic label for each pixel in the image satisfying that pixels with the same features would belong to a semantic object (as shown in Figure 2.1). Basically, a segmented image would be represented as a segment map.

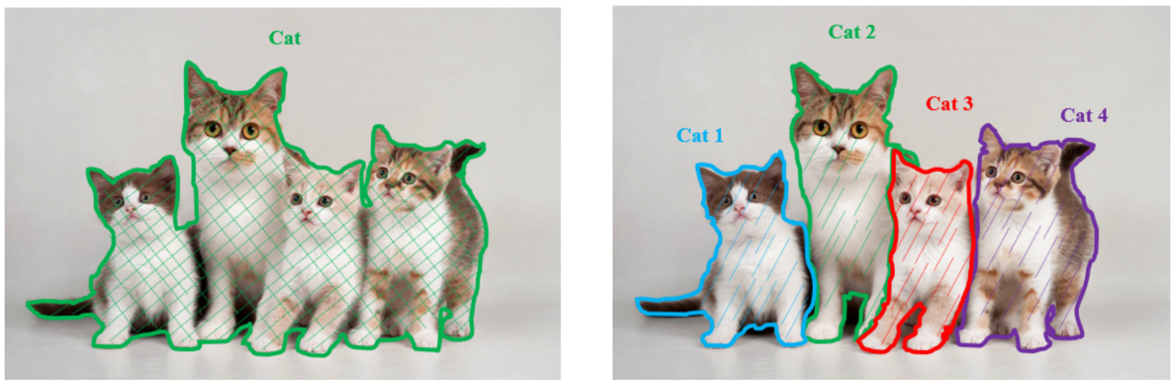

Figure 2.2: Results of semantic image segmentation and instance segmentation. Semantic segmentation on the left segments all the cats as one semantic class; instance segmentation on the right outputs each cat as a separate semantic segment.

Image segmentation is divided into two categories (as examples in Figure 2.2):

1. Semantic image segmentation: objects belonging to one semantic class is assigned the same label.
2. Instance segmentation: objects belonging to one semantic class is assigned as different instance label in that class.

A brief comparison of tasks in computer vision. Table 2.1 distinguishes fundamental tasks in computer vision to build the basic premise of the thesis.

Table 2.1: A brief comparison of tasks in computer vision.

	Target	**Input**	**Output**
Image Classification	Assign label to one object inside the image	Image containing one object + Labels list	Label of the object
Object Detection and Localization	Assign label to objects inside the image and draw bounding boxes	Image containing one/many objects + Labels list	Labels of objects in the image + Bounding boxes
Image Segmentation	Assign labels to each pixel in the image	Image containing one/many objects + Labels list	Label of each pixel in the image

2.3.2 Fully Convolutional Networks

Motivation. In the early stage of computer vision, semantic image segmentation was solved based on handcrafted features. The drawback of this method was the lack of semantic information, so that the segmentation performance was poor. Until the era of deep learning, tasks in computer vision were basically managed based on CNN. In 2015, J. Long et al [31] proposed an idea of applying fully connected network from VGG-Net [41] to the task of semantic image segmentation.

Network Architecture. Fully convolutional networks - FCNs [31] proposed by J. Long et al since 2015 was one of the very first CNN architecture to solve the problem of semantic image segmentation. Motivated by fully connected network from VGG-Net [41], FCNs is not a complex architecture (Figure 2.3) but shows the robust effectiveness to perform pixel-wise segmentation task. Below is the details of what FCNs did to handle the task of semantic image segmentation.

From image classification to semantic image segmentation. Normally, with a CNN-based image classification model, an input image passes through convolution layers for feature extraction. Then, those feature maps go through fully connected layers to be classified and assigned label. So, if we change those fully connected layers into 1×1 convolution layers **and** the input image is not suffered from downsizing process, the output is not a single label. An input image with size $H \times W$ is downsampled to $H/32 \times W/32$. At that position, if we perform upsampling process with a value of 32

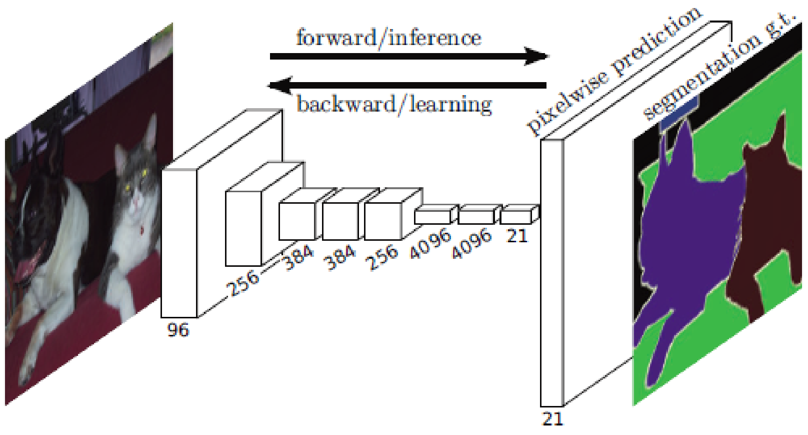

Figure 2.3: Fully Convolutional Networks [31] proposed by J. Long et al in 2015.

times of the feature map, we could receive a label map (or heatmap) of every single pixel whose size is equivalent to the input image.

Upsampling via transposed convolution. FCNs uses transposed convolution to upsample the size of the final feature map, below is an example of how to perform transposed convolution on 2D matrices (Figure 2.4).

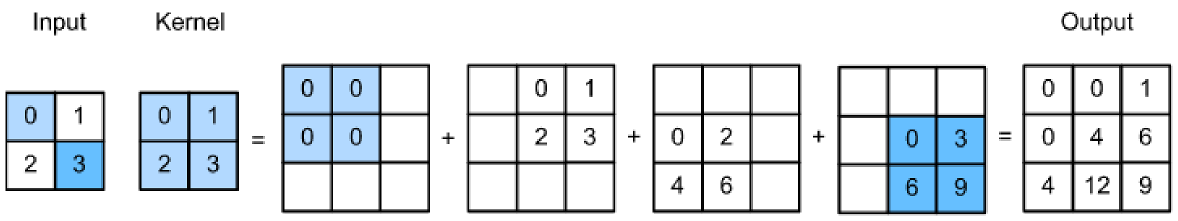

Figure 2.4: Upsampling example using transposed convolution. The input and kernel are 2×2 matrices, transposed convolution is performed with $stride = 1$. Multiplying the kernel with each component of the input, we get four 4×4 matrices. The output is the sum of those four matrices.[1]

FCNs variances by fusing outputs. Supposed 32 times upsampling is performed at $pool5$ where the feature map has the size of $H/32 \times W/32$, we have a so-called FCN-32s. The output from $pool5$ is upsampled twice and fuse with $pool4$ then perform a 16 times upsampling to be a so-called FCN-16s. Similarly, we have FCN-8s when using $pool3$. Figure 2.5 illustrates the process of forming FCNs variances.

[1]http://d2l.aichapter_computer-visiontransposed-conv.html

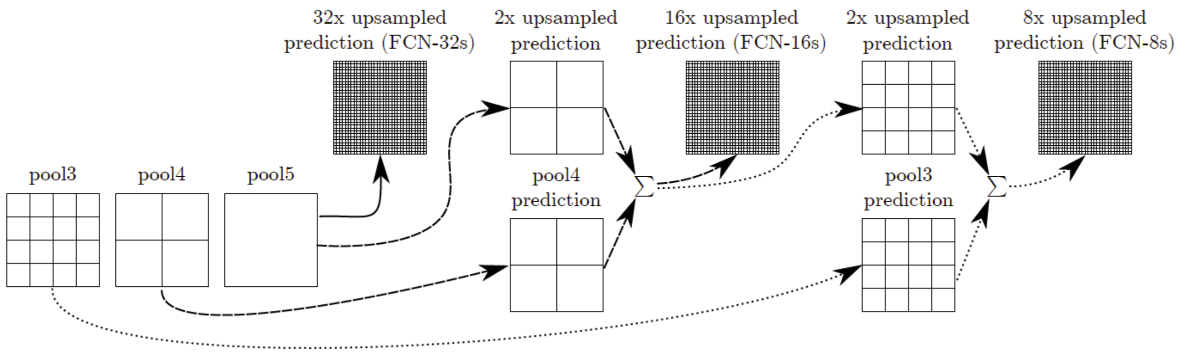
32x upsampled prediction (FCN-32s)
2x upsampled prediction
16x upsampled prediction (FCN-16s)
2x upsampled prediction
8x upsampled prediction (FCN-8s)
pool3
pool4
pool5
pool4 prediction
pool3 prediction
∑
∑

Figure 2.5: Fusing output to establish three variances of FCNs [31].

Conclusion. The experimental results yielded FCN-8s gave the best performance among the three variances of FCNs. Figure 2.6 is a typical instance of how different the segmentation results are. The segmented objects of FCN-32s looks coarse meanwhile FCN-16s and FCN-8s gained better segmentation performance due to the combination of multi-layer information.

FCNs is one of the very first architecture that applies CNN to solve the problem of semantic segmentation. This model is simple to implement and shows quick prediction (175ms on PASCAL VOC 2011/12). This used to be state of the art model at the time of publishment. However, drawback of this model is the loss of spatial information in upsampling stage.

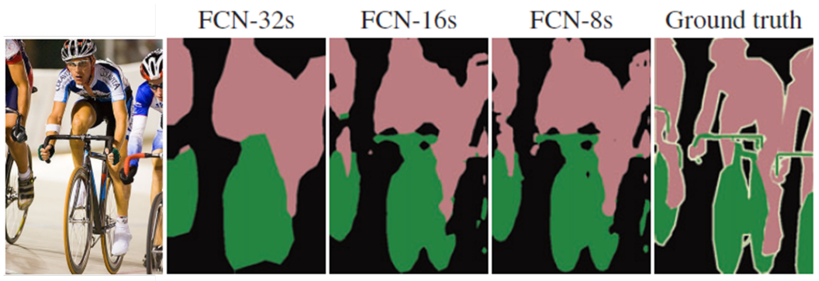

Figure 2.6: Differences among the three variances of FCNs with an exemplary image from PASCAL VOC 2012 [31].

2.3.3 Deconvolutional Network

Motivation. FCNs is a simple model yet gains positive results in the early stage of applying CNN to solve segmentation task. Nonetheless, there exists disadvantages of this model. For instance, to deal with too large or too small objects, FCNs showed mislabel cases (as in Figure 2.7). This is the result of lacking the spatial information of objects in the image. To solve the problem of FCNs, in the same year of 2015, H. Noh et al proposed the idea of a deconvolutional network (a.k.a. DeconvNet) [35] to segment objects.

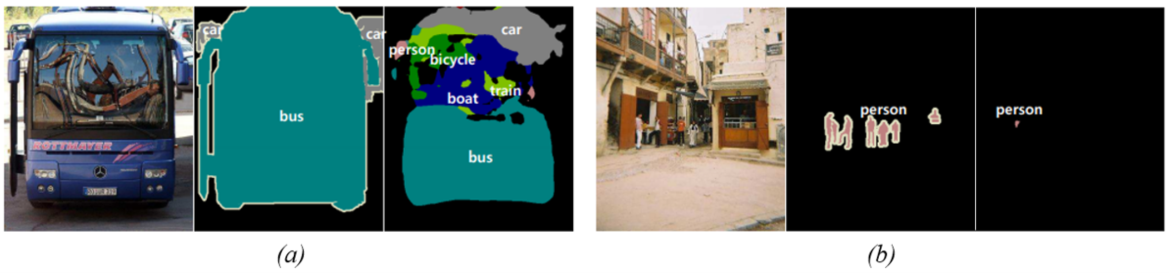

Figure 2.7: Instances of the drawbacks of FCNs model [35]. (a) Too large objects face over segmentation issue; (b) Too small objects is missed.

Network Architecture. Inherited the architecture from FCNs, DeconvNet is a combination of two components: encoder and decoder. Similar to FCNs, encoder plays role of a feature extractor. This encoder consists of convolution layers and pooling layers. On the other hand, decoder is the main improvement of this network. Instead of upsampling the output feature map to create segment map like FCNs, DeconvNet gradually carries on upsampling via deconvolution layers and unpooling layers. Details are in Figure 2.8. Backbone of the network is VGG without fully connected layers. The output is steadily recovered thanks to unpooling layers in decoder. Unpooling, besides storing the value of the feature maps, also has to remember the location of that value. Figure 2.9 illustrates the operations of pooling and unpooling.

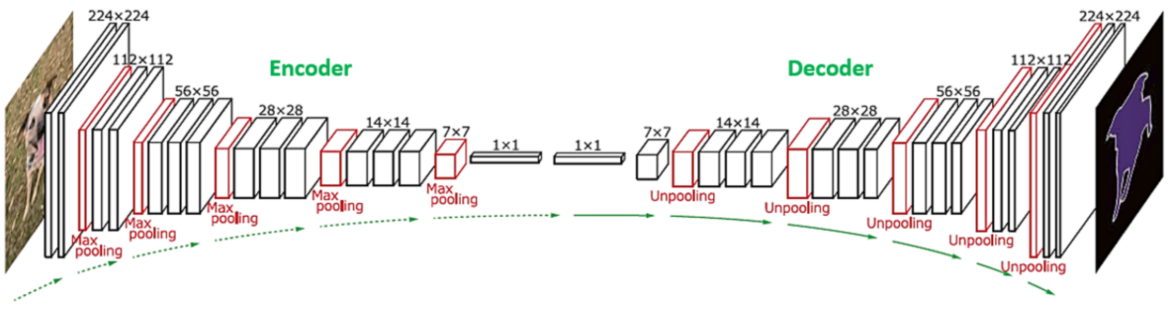

Figure 2.8: Deconvolutional Network [35] Architecture.

To diminish the limitation of FCNs, 50/2000 regions of proposal (bounding boxes) are selected in each input image by using Edgebox algorithm. DeconvNet is applied to each region and the results then is combined together to form an output segment map with the same size as the input image. This method helps solve the problem of multi-cale objects. Nevertheless, it is noticed that these regions of proposal are selected without semantic concept.

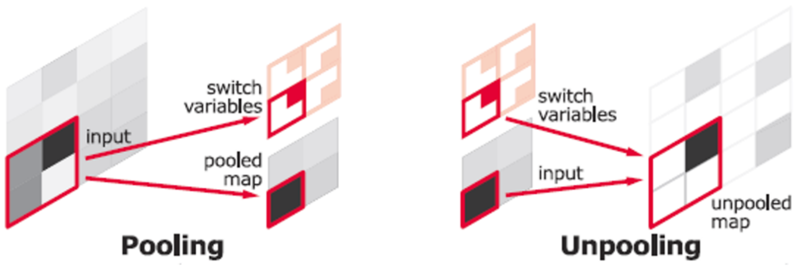

Figure 2.9: Illustration of pooling and unpooling in DeconvNet [35].

Conclusion. Figure 2.10 displays the differences between the results of FCNs and DeconvNet on the same images presented in heat map mode. As can be seen, the objects in those images are better segmented with DeconvNet, which is the result of gradual information recovery via deconvolutional network. FCNs has the ability of extracting the overall shape of the objects, meanwhile, DeconvNet can capture better details of objects in the image and solve multi-scale issue. H. Noh et al proposed to fuse the results of the two methods and achieved 72.5% mIoU on PASCAL VOC 2012.

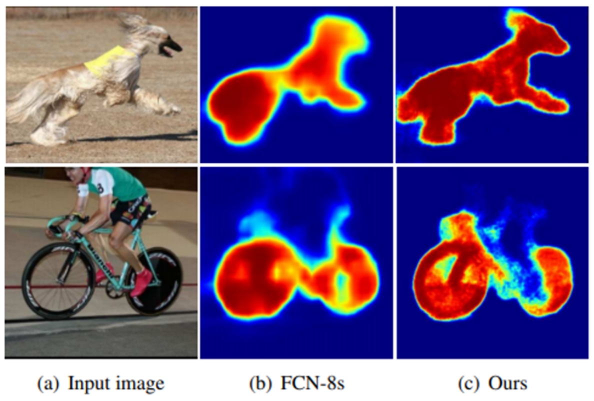

Figure 2.10: Comparison of the results of FCNs and DeconvNet [35].

The advantage of gradually performing deconvolution instead of upsampling once in FCNs is visualized in Figure 2.11. The author fused the strengths of both FCNs and DeconvNet to result in better segmentation performance. However, the maintaining drawback of those models is the lack of information when using pooling operation.

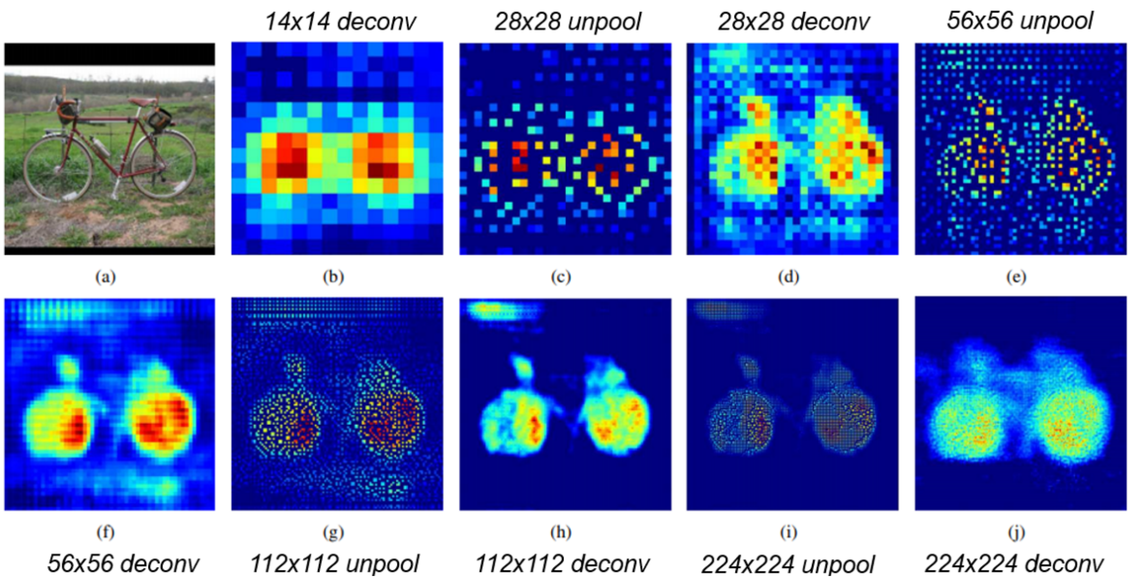

Figure 2.11: Visualization of decoder stage represented as heat maps. The information of the bicycle is gradually recovered in the last layers of DeconvNet [35].

2.3.4 U-Net

Motivation. As can be seen from the previous sections, semantic segmentation models of FCNs and DeconvNet have encoder-decoder architecture in common. Encoder takes the responsibility of extracting image features and decoder recovers information to form output segment map. To leverage the spatial information, O. Ronneberger et al proposed U-Net [39]. The speciality of this network is the appearance of skip-connection to combine feature at different levels. In the beginning, U-Net aimed at solving bio-medical image segmentation whose properties are the label imbalance among training data and the requirement of high accuracy. Later, this U-Net is applied to other fields of segmentation. Recently, Unet has another version of neural architecture search [45] to solve medical image segmentation.

Network Architecture. Based on the appearance of the visualization of the network, it is named U-Net (Figure 2.12). The network is a combination of two components: encoder and decoder. Particularly, the function of encoder is extracting image features, each block of the encoder consists of without-padding 3×3 convolution layers, ReLU activation function and max pooling operations. This encoder has a typical architecture of a convolutional neural network. The rest of U-Net is decoder part whose function is to recover the information from feature maps to output label maps. Decoder includes convolution layers in correspondence with those of encoder. Each block has a 2×2 (up-conv 2×2) convolution and concatenation with information from encoder stage

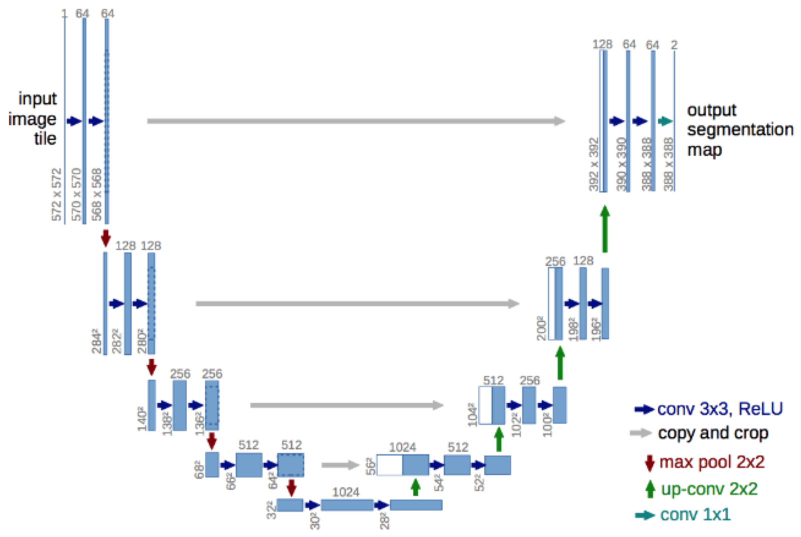

Figure 2.12: U-Net Architecture [39].

(skip-connection), then there are two 3×3 convolution layers and ReLU activation function.

Besides the mission of upsampling feature maps, the decoder also performs concatenation with information from layers in encoder stage. The concatenation process is applied on corresponding layers among the two stages. Those connection helps recover the information lost due to pooling operation. Note that instead of passing the whole image into the network and using sliding window, U-Net separates the image into pieces and each piece goes into the network. Thus, the image size is not a big problem in this architecture. Figure 2.13 illustrates that a result region is affected by a wider region before, which support to capture surrounding information.

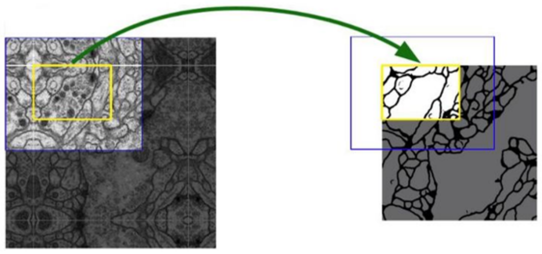

Figure 2.13: Extracted features are the results of a wider region [39].

Conclusion. The proposed U-Net achieved top results on bio-medical image segmentation challenges: EM segmentation challenge (from ISBI 2012), ISBI cell tracking challenge 2015. U-Net proved to be suitable for many segmentation tasks with better executing time compared with methods using sliding window. Moreover, U-Net also works well with less training data. However, the drawback of this network is time-consuming when training.

2.3.5 Pyramid Scene Parsing Network

Motivation. It is widely believed that the spatial information of an image is really important to solve the issue of semantic image segmentation. Scene parsing is basically segmentation problem but attention is paid more to the overall of the scene in the image. The more spatial information can be got the better accuracy of the segmentation system is claimed. In 2017, Pyramid Scene Parsing Network (PSPNet) [53] is proposed with an aim at focusing on leveraging the spatial information of the image to serve scene parsing problem.

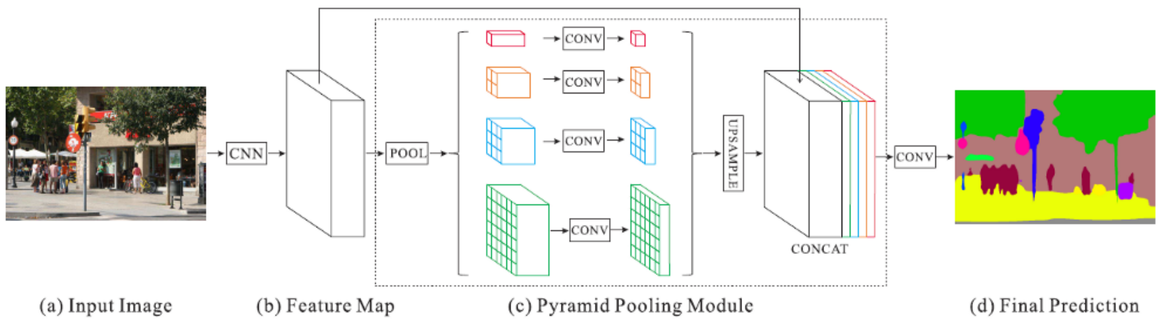

Figure 2.14: Architecture of Pyramid Scene Parsing Network [53].

Network Architecture. Figure 2.14 illustrates the process of passing an image into PSPNet. Particularly, input image (a) passes through a CNN to extract feature maps (b), those maps are processed by pyramid pooling module (c) to exploit spatial information, the results then are combined to predict labels of pixels in the input image (d). PSPNet concentrates on the global information of the image. The most important component in this network is the pyramid pooling module (see component (c) in Figure 2.14).

Fistly, a CNN is used to extract information of the image. Here, PSPNet architecture uses ResNet [14] as its backbone with dilated convolution to do features extraction (similar to the idea of atrous convolution in DeepLab [2, 3, 4, 5]). Output of this stage is feature maps with the size of 1/8 the original input size.

Then, those feature maps go through pyramid pooling module to exploit global spatial information. In details, the used operations are global average pooling with 2×2, 3×3 and 6×6 convolution layers. To control the depth, PSPNet uses 1×1 convolution. There are N size of convolution to extract features ($N = 4$ in Figure 2.14c). The depth of feature maps in this step would be $1/N$ compared with maps at (b). Bilinear interpolation is used to upsize the feature maps after pooling. The maps are concatenated together to form labels map output.

Conclusion. PSPNet achieved best performance when published on PASCAL VOC 2012 (85.4% mIoU) and CityScapes (80.2% mIoU), outperformed previous models like FCNs, DeepLabv1 and DeepLabv2. Figure 2.15 proves how efficient PSPNet segments objects on PASCAL VOC 2012 compared with baseline FCNs (backbone Resnet50).

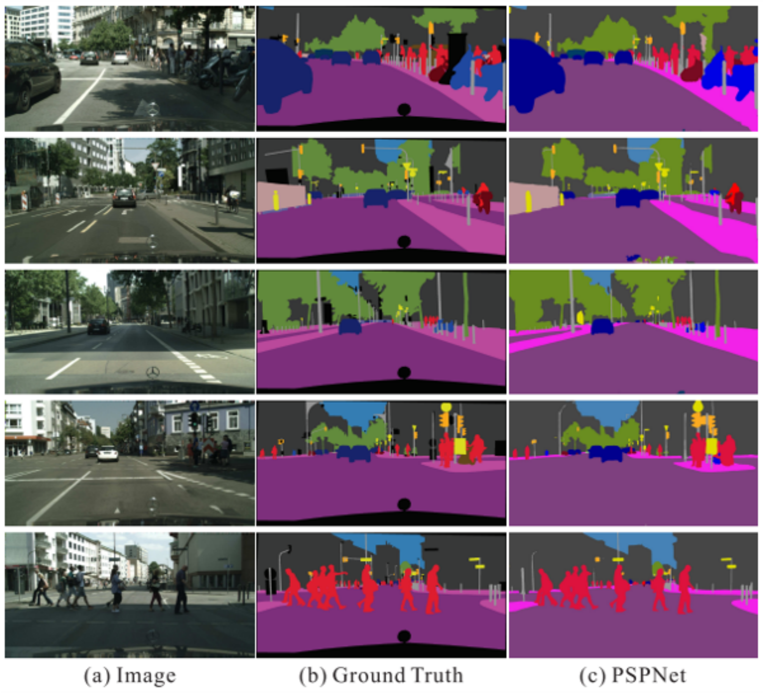

Figure 2.15: Results of Pyramid Scene Parsing Network on Cityscapes benchmark [53].

PSPNet exploits global spatial information better than other architectures and leverages the connection among objects in the image thanks to pyramid pooling module. However, drawback of this network is the limitation of FCNs backbone which cannot exploit the overall image information.

2.3.6 RefineNet

Motivation. So far, deep convolutional neural networks have proved theirs ability to handle the problems of computer vision, specially semantic segmentation. However, there is a typical problem that related to network architecture. In details, pooling or striding convolution lead to the loss of visual information after layers. This causes the segmentation result at the positions of boundaries or details not clear.

Many proposed solutions have been conducted to tackle the mentioned problem but there are still limitations. Particularly, three following solutions are mentioned to be typical instances. Deconvolution (a.k.a. transposed convolution) is capable of gradually recovering the information from feature maps. Yet, low-level features are hardly recovered as they are lost in down-sampling stage. Dilated convolution (a.k.a. atrous convolution) can capture better fields of view with the same amount of parameters and it is not neccessary to do down-sampling.

However, atrous convolution is not only computational expensive but also require more GPU resources. The reason is that the operation is performed on a large number of high-resolution feature maps. Another solution is skip connection which helps to combine low-level features from the previous stage. But a disadvantage of this skip connection is the lack of spatial information. Understand how important both low-level and high-level features are, RefineNet [25] exploits multi-level features to output a high quality segmentation map.

Network Architecture. RefineNet uses Resnet as a backbone to extract image information. Figure 2.16 illustrates the overall framework of RefineNet, (a) yields the overview of the network, meanwhile others (b, c, d) show details of each component. Firstly, different resolutions of feature maps go through residual convolution unit (RCU). This stage is a so-called cascaded architecture. RCU (b) is a residual block with a change of removing batch normalization layers. Next in the fusion step (c), multi-resolution fusion is done to merge the feature maps using element-wise summation. In chained residual pooling module (d), the output feature maps of all pooling blocks are fused together with the input feature map via summation of residual connections. It aims at capturing background context from a large image region. Finally, another RCU is placed to generate output segment maps to employ non-linear operations on the multi-path fused feature maps.

Conclusion. RefineNet proves its performance on various benchmarks such as NYUDv2, CityScapes and Pascal VOC 2012 with the IoU scores of 46.5%, 73.6% and 83.4%,

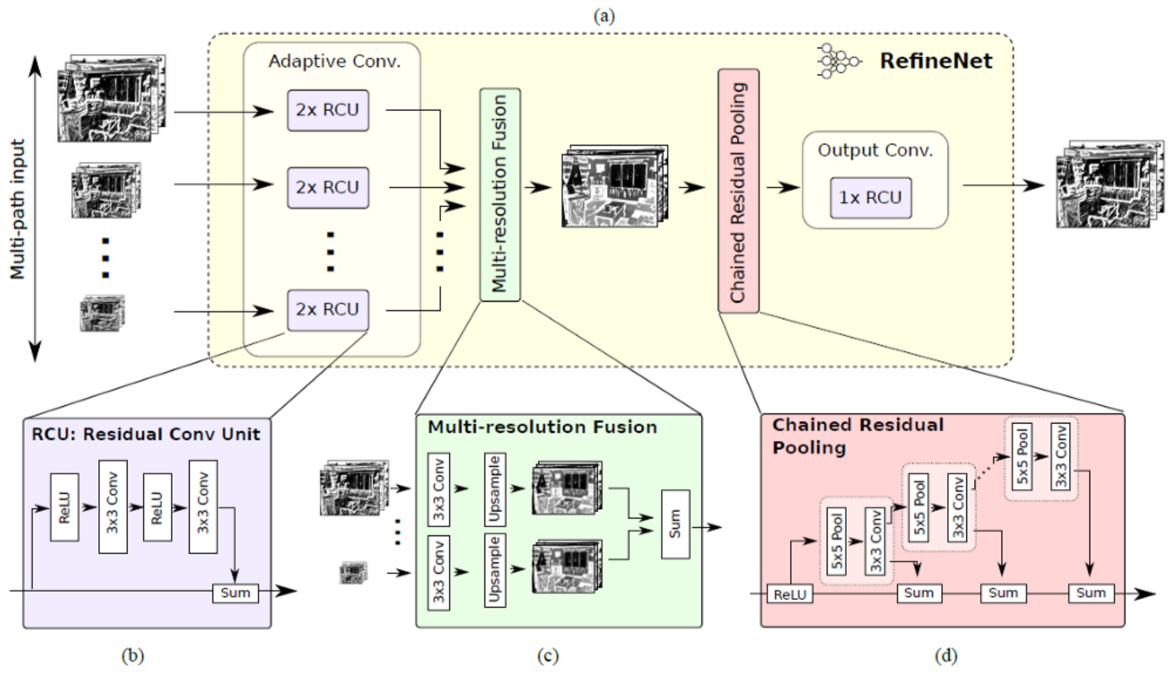

Figure 2.16: Overall Architecture of RefineNet Model [25].

respectively. To conclude, RefineNet introduces a novel multi-path refinement network for semantic segmentation. The cascaded architecture combines high-level and low-level features to generate high-resolution segmentation maps effectively.

2.3.7 Google DeepLab Family

From 2018 to the middle of 2020, models that yield high performance on semantic image segmentation are the family of Google DeepLab (evaluated on PASCAL VOC 2012 benchmark). Published for the first time since 2016, there are four versions of this model (v1, v2, v3 and v3+). The main points of those networks are listed below:

- DeepLabv1 [2] proposed atrous convolution to control the resolution of the extracted feature maps with a hyperparameter of dilated rate, this atrous convolution has the ability of gaining a better field of view but maintaining the same number of parameters; another contribution is to recover object boundaries with conditional random fields (CRFs).

- DeepLabv2 [3] combined atrous convolution with pooling operation to come up with a so-called atrous spatial pyramid pooling module (ASPP) to maintain aspect ratio of objects in the image; kept using CRFs to recover object boundaries.

- DeepLabv3 [4] applied image-level features to ASPP module and used batch normalization to better train the model.
- DeepLabv3+ [5] extended DeepLabv3 by adding a decoder module to improve the segmentation result, focused on object boundaries.

Here we presents key features of DeepLabv3+, which is considered as the finest version of DeepLab family.

Motivation. Realizing that the encoder-decoder architecture has the ability to recover spatial information and the module of ASPP can capture multi-scale information, L.-C. Chen et al proposed to combine these two into a so-called model DeepLabv3+ [5]. Moreover, to solve the problem of computational cost, the authors proposed to use atrous separable convolution. DeepLabv3+ promised to bring about better semantic segmentation results.

Network Architecture. There are four key factors that lead to the success of DeepLabv3+. The details are mentioned below:

Atrous convolution. In CNN architecture, the information extraction of data is done with convolution layers. To capture semantic information in multi-scale spatial levels, we try to apply convolutions with different sizes (3×3, 5×5 for instances) and the results are the increase in the number of parameters and computational cost. Atrous convolution (a.k.a. dilated convolution) was introduced to cope with this problem.

With the same number of parameters, atrous convolution provides a wider field of view, which helps to better capture spatial information of the image. Figure 2.17 illustrates the differences between atrous and standard convolution, (this example is a process of convolution with $stride = 1$ and $pad = 0$). In details, with the same output size of 3×3, standard convolution receives information from a 5×5 area, meanwhile atrous convolution scans a wider 7×7 area.

Besides, we can adjust the field of view of atrous convolution via a so-called dilated rate r parameter. In Figure 2.17, atrous convolution is used with $r = 2$, which means there exists zero padding inside the kernel to represent the dilation. These positions raise no multiplications or parameters. The assumption in DeepLabv3+ is that not all the pixels which stay next to each other yields the same level of importance. Therefore, we can leverage the information from pixels which are far away from each other to have

[2]Image source: medium.com

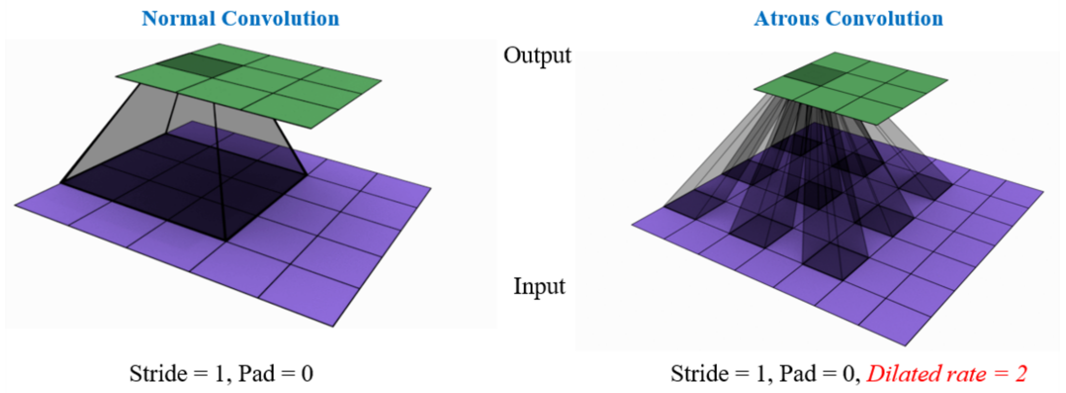

Figure 2.17: Illustration of standard and atrous convolution. Atrous convolution gives a wider field of view with the same number of parameters compared with standard convolution.[2]

a more abstract and general view of the concerned object. We can adjust arbitrary field of view by controlling dilation rate r. Standard convolution can be concerned as a special case of atrous convolution with rate $r = 1$.

Let's take an example with the assumption that input is one-dimension data, atrous convolution is defined as the following Equation 2.1. For interested readers, please refer to [3] for more details.

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k]w[k] \tag{2.1}$$

where,

- $x[i]$: one-dimension input data
- $w[k]$: filter with kernel K
- r: dilated rate of atrous convolution
- $y[i]$: output of atrous convolution

The core properties of the two kinds of convolution is presented in Figure 2.18. Atrous convolution is applied to a part of the input then upsampling operation is used to recover its size. The loss of information occurs during the downsampling process. Meanwhile, atrous convolution solve the problem and even can capture wider field of view.

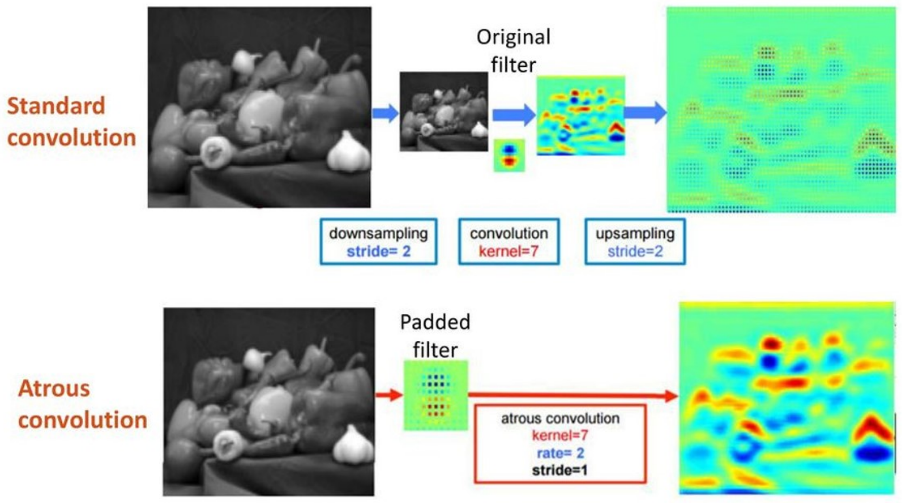

Figure 2.18: Illustration the differences when using standard and atrous convolution ($r = 2$). When applying standard convolution to the input image, we downsize the image to $H/2 \times W/2$ then apply filter 7×7. In contrast, atrous convolution is directly applied to $H \times W$ input image. Basically, when using standard convolution, we process on 1/4 of the input image then upsize the feature map in output. Atrous convolution is applied to the whole image, therefore results in more dense feature maps.

Encoder-decoder architecture. DeepLabv3+ uses encoder-decoder architecture (Figure 2.19), each module in the model is responsible for a separate purpose. Particularly, encoder module extracts context information of the image by using ASPP module with different rates of atrous convolution. Meanwhile, decoder module recovers the information of the image and outputs a label map. The main contribution of DeepLabv3+ is the decoder module while the encoder leverages DeepLabv3. In decoder stage, the model concatenates feature maps of high level and low level with an aim at capturing semantic information. Thus, DeepLabv3+ can perform more accurate segmentation, specially along object boundaries.

Atrous spatial pyramid pooling module. In DeepLabv3, there are two choices of using cascading or parallel (ASPP) module. However, DeepLabv3+ decided to use only parallel module to extract features. ASPP module (Figure 2.20) consists of a 1×1 convolutional layer and three layers of atrous convolution with dilated rate $r = 6, 12, 18$ to extract multi-scale semantic information and a global average pooling to gain abstract context information.

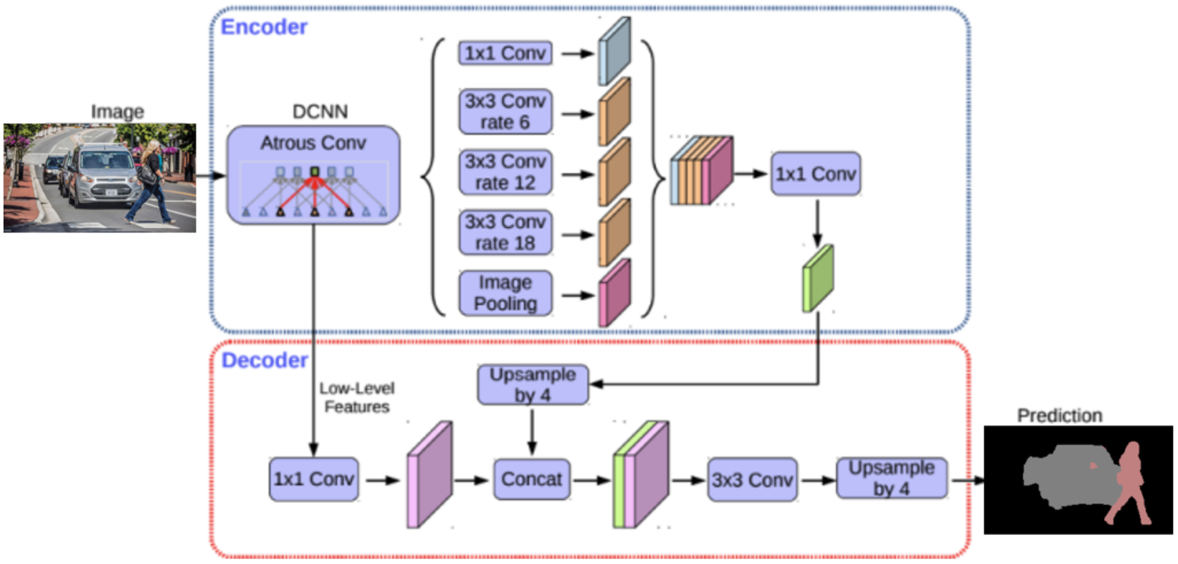

Figure 2.19: DeepLabv3+ architecture [5].

Atrous separable convolution. Atrous separable convolution is a technique that helps reduce computational cost compared with standard convolution. As its name, atrous separable convolution is a combination of two parts: depthwise and pointwise. Let's take an example of atrous separable convolution following this description: supposed we have an input $12 \times 12 \times 3$ feature map, then apply three $5 \times 5 \times 1$ kernels on it to output a $8 \times 8 \times 256$ feature map.

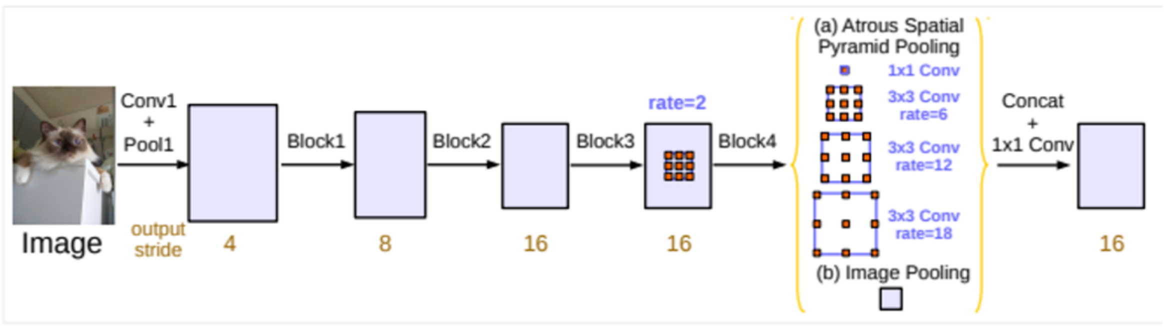

Figure 2.20: Atrous spatial pyramid pooling module in DeepLabv3 [4].

In standard convolution (Figure 2.21), we have to do convolution $5 \times 5 \times 3$ a number of 256 times to meet the above output requirement. The number of parameters is $5 \times 5 \times 3 \times 256 = 19,200$ (parameters). The number of multiplications is $5 \times 5 \times 3 \times 256 \times (8 \times 8) = 1,228,800$ (multiplications).

Meanwhile, separable convolution (Figure 2.22) performs two separate steps. In

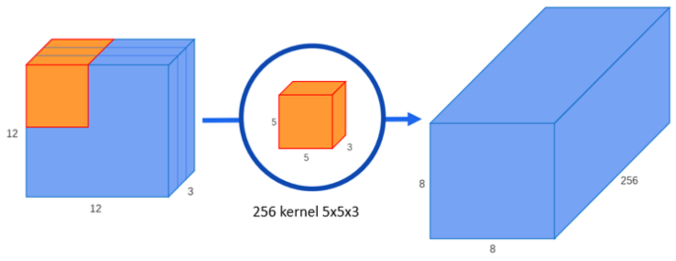

Figure 2.21: Visualization of normal convolution.

depthwise step, the $5 \times 5 \times 3$ kernel is applied to the input feature map to output a $8 \times 8 \times 3$ map. Then in pointwise step, we perform 256 times of 1×1 convolutions to control the output depth. The number of parameters is $(5 \times 5 \times 3) + (1 \times 1 \times 3 \times 256) = 843$ (parameters) compared to $19,200$. The number of multiplications is $(5 \times 5 \times 3 \times 8 \times 8) + (1 \times 1 \times 3 \times 256 \times 8 \times 8) = 53,952$ (multiplications) compared to $1,228,000$. To sum up, the number of both parameters and multiplications in separable convolution reduces 22 times compared with standard convolution. Basically,

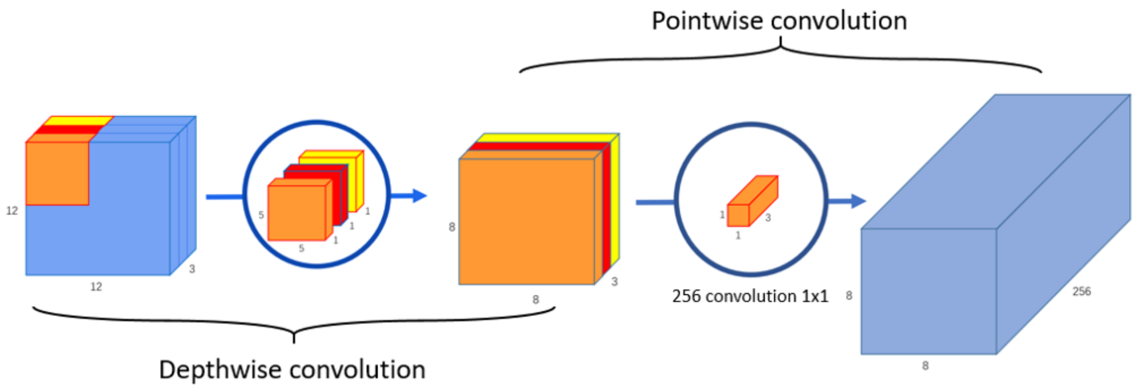

Figure 2.22: Visualization of separable convolution.

standard convolution transforms the input feature map 256 times by 256 $5 \times 5 \times 3$ kernels. Whereas, separable convolution transforms the input once in depthwise step and pointwise step is responsible for transforming the result of depthwise step 256 times by 256 1×1 convolution blocks

Moreover, DeepLabv3+ uses separable convolution with an implementation of atrous convolution in depthwise step and its rate is $r = 2$ (Figure 2.23). This both leverages the ability of atrous convolution and reduces the computational cost. DeepLabv3+ replaced standard convolutions with atrous separable convolutions. Yet, atrous separable convolution still have disadvantages. Particularly, this technique does not help in models with few parameters as the reduction of parameters in atrous separable convolution may harm the training process.

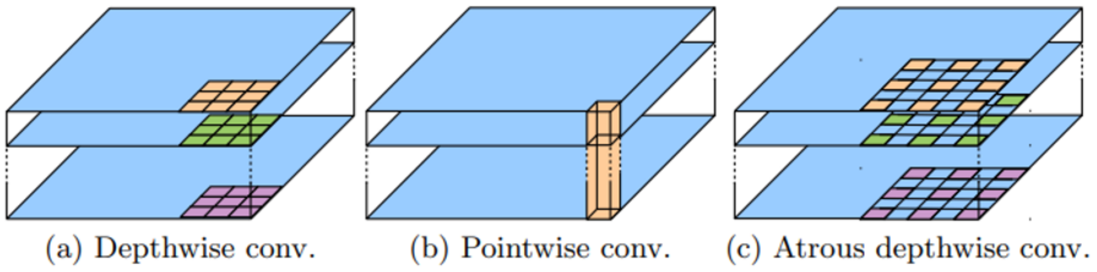

Figure 2.23: Visualization of atrous separable convolution [5].

Discussion. To sum up, there are four main contributions in DeepLabv3+ model. (i) DeepLabv3+ used encoder-decoder architecture in which the encoder originates from DeepLabv3, and the novel decoder combined the high-level information and low-level information; (ii) used atrous convolution to trade off precision and runtime; (iii) leveraged atrous convolution for both ASPP module and decoder module; (iv) yielded state-of-the-art result on PASCAL VOC 2012 and Cityscapes.

2.4 Generative Adversarial Network

2.4.1 Overview of Generative Adversarial Network

Generative Adversarial Network (GAN) [11] is a class of machine learning frameworks invented by Ian Goodfellow et al in 2014 in NIPS. GAN opens a new innovative horizon of deep learning, particularly computer vision. To be specific, GAN is an approach of generative model using adversarial methods. Adversarial learning is a technique of machine learning that try to fool models by providing deceptive data. In recent years, many applications were created based on GAN.

Firstly, generated datasets can be used for multiple purposes (as in Figure 2.24). In this case, GAN is used to create new samples from available dataset. For example, new plausible handwritten digit dataset (MNIST [22]), small object photograph dataset (CIFAR-10 [21]) and the other is Toronto Face Database (TFD [37]).

Secondly, generating photographs of human faces is also a significant achievement of GAN. Tero Karras et al [19] in 2017 illustrated the plausible realistic photographs

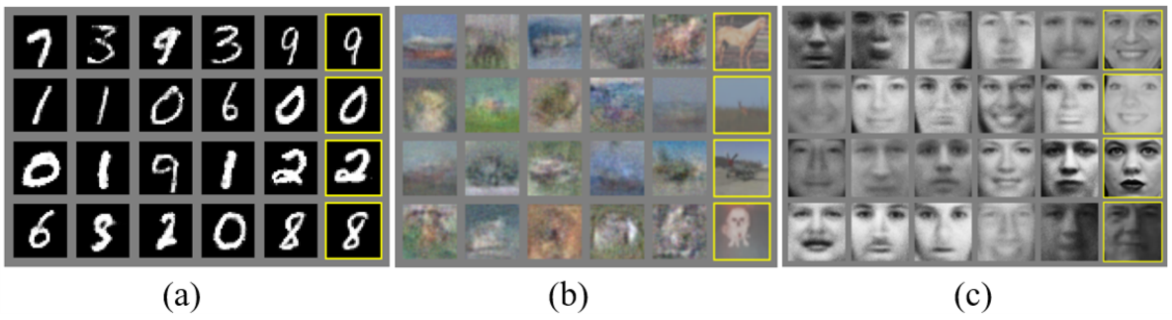
Figure 2.24: The application of GAN to generate datasets. New example images in datasets are generated by GAN. (a) MNIST handwritten digit dataset, (b) CIFAR-10 small object photograph dataset, (c) Toronto Face database.

of human faces. Beside generating faces, they also modify faces by age, makeup or complexion which contribute to create a whole face. Therefore, GAN is attracted by netizen (social network users) especially younger generations for the time being.

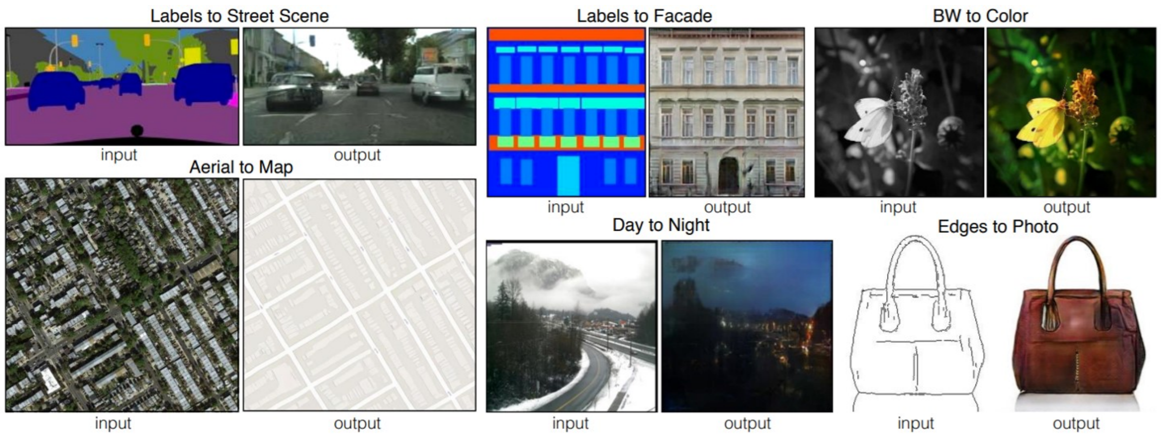

Figure 2.25: Applications of image to image translation [16]. Image translation based on paired dataset such as day-to-night, sketch-to-image, segment map-to-photo and so on.

Thirdly, image-to-image translation [16] is one of the most attractive brands of applications of GAN. There is a vast number of domains for image-to-image translation usages. Particularly, translation of semantic images to photographs of cityscapes and buildings, translation of photos from day to night, translation of sketches to color photographs (Figure 2.25), translation from summer to winter, translation from photographs to artistic painting style and so on as shown in Figure 2.26.

Architecture. A simple generative adversarial network is a combination of two CNNs (Figure 2.27):

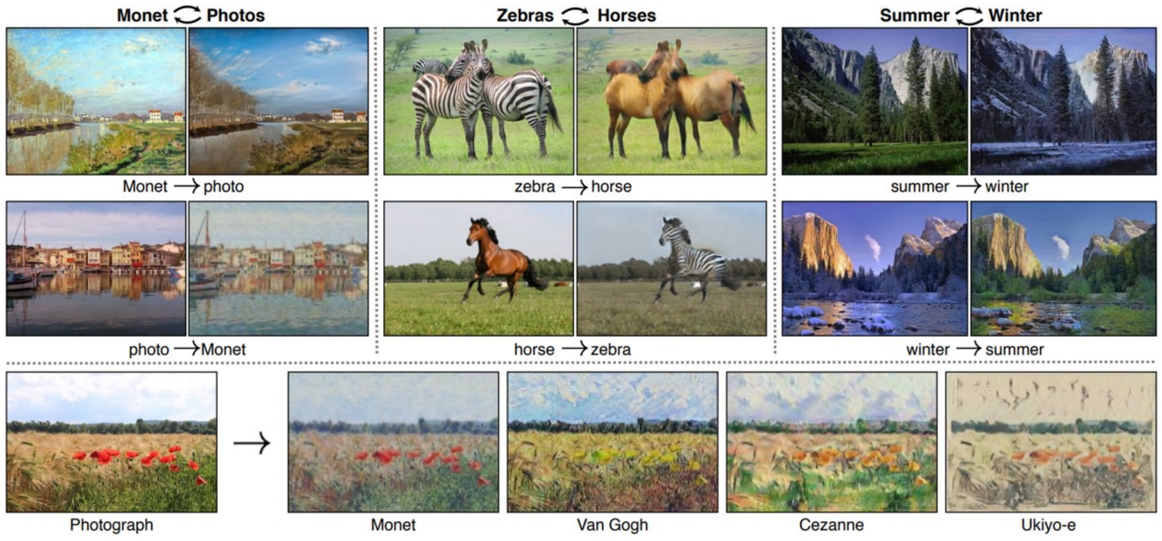

Figure 2.26: Applications of image to image translation [54]. Various domains are translated using unpaired image-to-image translation methods such as Monet-to-photo, zebras-to-horses, summer-to-winter and so on.

- **Generator**: learn to generate fake data which has the same distribution as provided real one.
- **Discriminator**: learn to distinguish the generated fake data and real data. Discriminator tends to penalize the generator for creating implausible data.

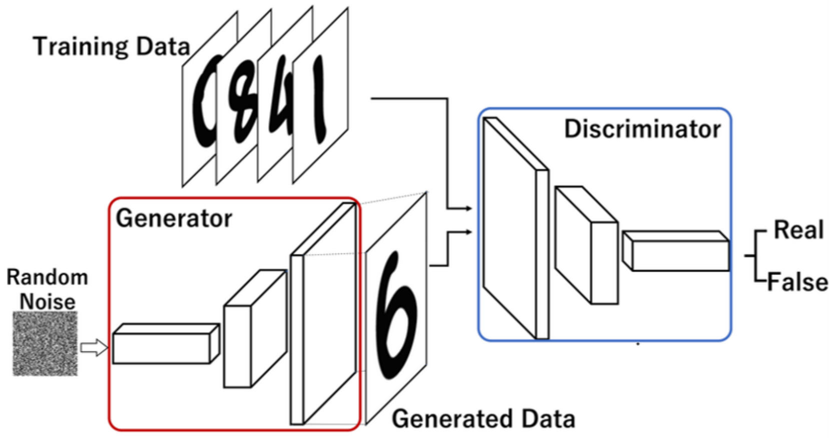

Figure 2.27: An overview framework of GAN which contains the generator model G and the discriminator model D.

Loss Function. When it comes to discriminate the real and the fake samples, 0 or 1, we first come up with binary cross entropy as a loss function. Particularly, binary cross entropy is formed in the Equation 2.2. y denotes ground truth and $\hat{y}$ denotes predicted result.

$$\mathcal{L}(y, \hat{y}) = min[-y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})] \tag{2.2}$$

Taking mini-max game for instance, mission of classifier is discriminating real data and fake data. Therefore, discriminator D has two formulas for real (Equation 2.3) and fake (Equation 2.4):

- Consider $y = 1$ then Equation 2.2 becomes Equation 2.3 with input real image x:

$$\mathcal{L}(y, \hat{y}) = min[-log(\hat{y})] = min[-log(D(x))] = max[log(D(x))] \tag{2.3}$$

- Consider $y = 0$ then Equation 2.2 becomes Equation 2.4 with input latent code z:

$$\mathcal{L}(y, \hat{y}) = min[-log(1 - \hat{y})] = min[-log(1 - D(G(z)))] = max[log(1 - D(G(z)))] \tag{2.4}$$

To be specific, $max[log(D(x))]$ helps to correctly label real images x to 1, while $max[log(1 - D(G(z))]$ tends to label fake images generated from G to 0. The opposite is true for generator G, so we have the loss function as illustrated in Equation 2.5. Instead of maximizing Equation 2.4, we minimize it.

$$\mathcal{L}(y, \hat{y}) = min[log(1 - D(G(z)))] \tag{2.5}$$

Training. The training progress of GAN is complex due to its two separable networks. A GAN model is evaluated on its two components: discriminator as well as generator. It is hard to identify when model converges. Hence, there are two phases to train a GAN, which is an **alternative** process: train discriminator and train generator. Repeating two phases above to continue to train the whole GAN model. To be more detailed, the generator is freezed when training the discriminator, by this way, the discriminator learns to figure out how to classify real and fake data. The opposite is true for training generator. The discriminator is constantly maintained during the training process of generator, which helps to generate realistic images having the most similarity with real data. As a result, GAN allows its components to *tackle* each other consistently, which not only improves classifier to predict more precise but also develops the generator to create more realistic images.

2.4.2 Conditional Generative Adversarial Network

Conditional GAN (cGAN) [32] is an improved version of traditional GAN [11], which is built via refining the generator. With an aim at controlling the output, a novel constraint is combined with the input as an additional information for the machine to learn more stable. This section briefly reviews the specifications of cGAN.

Motivation. Although GAN [11] model has significant achievement of generating new *random* samples, one problem is that **there is no way to control the types of images** that we target. All of images are randomly generated and there is no relationship between the latent space input to generator and the generated images. To tackle problem above, cGAN [32] was invented. The conditional generative adversarial network is a type of GAN that involves the conditional generation of images by the generator model. Therefore, image generation can be conditional generated on a class label, which was a result of added constraint. There are two motivations for making use of class label information in a GAN model. On the one hand, the authors improve the GAN training process stably by adding class labels. On the other hand, the target image would be conditional generated via the additional information.

Additional Input. Additional information is a type of prior knowledge that correlated with the input images, e.g. class label. It can not be only used to improve GAN but also come in form of more stable training and generated images that have better quality. Moreover class labels can also be used for the targeted or deliberate generation of images of a give type.

According to Figure 2.28, (a) The results of GAN were randomly generated which was a result of no relationship between. (b) The results of Conditional GAN were orderly generated that was targeted images of a given type class label.

Loss Function Almost formula is the same as normal GAN loss, however, the different one is the constraint y. Here in Equation 2.6, y work as a label or the condition that makes the model generate image depending on our purpose.

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(D, G) = E_{x \sim P_{data}(x)} \log(D(x|y) + E_{z \sim P_z(z)} \log(1 - D(G(z|y))) \qquad (2.6)$$

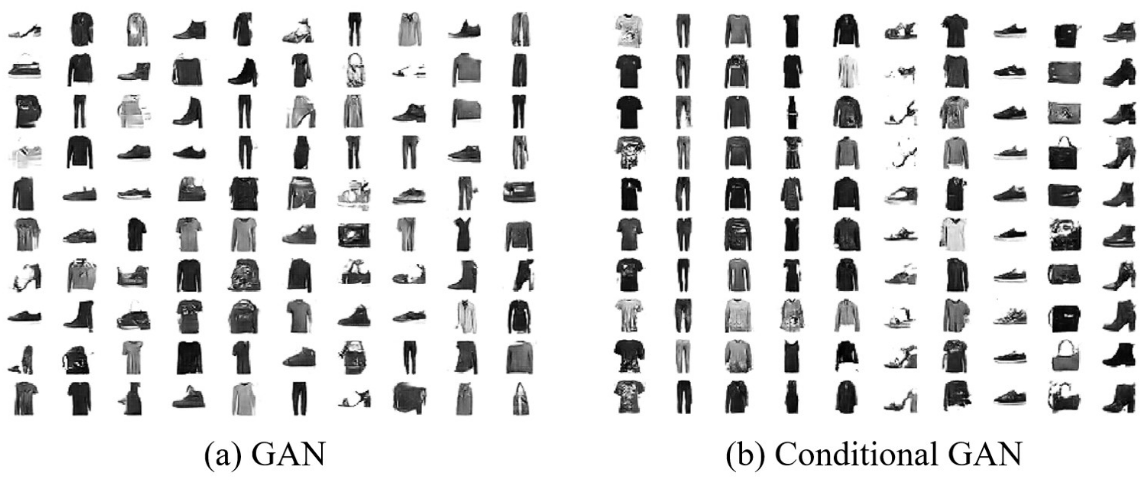

Figure 2.28: Comparison result between traditional GAN and Conditional GAN [32]. (a) is randomly generated images, (b) is conditional generated images.

2.4.3 Pix2Pix

Motivation. To rely on cGAN [32], Pix2Pix [16] is also an updated version of cGAN whose additional **information is an image instead of a class label**. In other words, Pix2Pix [16] is a Generative Adversarial Network designed for general purpose image-to-image translation. This section shows the overview of Pix2Pix.

Conditional GAN [32] has potential improvement when generating targeted or purpose images like MNIST dataset, MNIST fashion dataset or handwriting dataset. All mentioned datasets are simple and easy for generator to create plausible images. However, Pix2Pix model is more ambitious when applying on complex dataset with higher resolution as well as higher quality, especially Satellite2Map dataset or Cityscapes dataset.

Specification of Pix2Pix. To be more specific, this Pix2Pix model is a type of cGAN where the generation of output image is conditional on an input, in this situation, a source image. The discriminator receive a couple input, target image and sources image and should determine if the target is a plausible translation of source image.

Besides, the generator is updated via adversarial loss, which encourages to generate plausible images in destination domain. In addition, it is also trained via L1 loss between generated image and the ground truth image. This L1 loss encourages the generator model to produce more plausible translations of the source image.

Loss Function. The conditional-Adversarial Loss (Generator versus Discriminator) is formatted as follow:

$$\mathcal{L}_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,y}[\log(1 - D(x, G(x, z)))] \tag{2.7}$$

The striking feature is the addition of the loss calculating the distance between the real image and generated image. The L1 loss function previously mentioned is illustrated below:

$$\mathcal{L}_{L1}(G) = E_{x,y,z}[||y - G(x, z)||_1] \tag{2.8}$$

Combining these functions $L_{cGAN}(G, D)$ and $L_{L1}(G)$ in Equation 2.7 and 2.8 with hyper-parameter λ that results in:

$$G = arg \min_G \max_D \mathcal{L}_{cGAN}(D, G) + \lambda \mathcal{L}_{L1}(G) \tag{2.9}$$

2.4.4 CycleGAN

Motivation. Image-to-image translation has exponential and significant achievement. However, **the biggest challenge is about dataset**, especially pair dataset. Almost mentioned GAN model require dataset or paired images. Take the translation from satellite to map for instance, if we are excited at translating these images, we have to create a manual training dataset involving couple images. It is extremely costly and labours-consuming for each domain translation. Therefore, CycleGAN was developed to deal with this dataset problem.

Generally speaking, CycleGAN is based on GAN, but it does not require a dataset of paired images. The advantage of CycleGAN is that it can be trained without paired examples. Instead, this model could use a collection of images from each domain and extract features and style as well of images in the collection to perform the translation.

Specification of CycleGAN. To be more specific, the CycleGAN model architecture is comprised of two GAN models:

- Generator A and Discriminator A
- Generator B and Discriminator B

The Generator models work on image translation, which is conditional on an input image, in this case, image from the other domain. Generator A takes an image from domain B as an input and Generator B takes an image from domain A as an input:

- Domain B $\rightarrow$ Generator A $\rightarrow$ Domain A
- Domain A $\rightarrow$ Generator B $\rightarrow$ Domain B

Each Generator has a corresponding Discriminator model. The Discriminator A takes real images from domain A and generated images from Generator A, then predict if they are real or fake. The same is true for Discriminator B:

- Domain A $\rightarrow$ Discriminator A $\rightarrow$ Real/Fake
- Domain B $\rightarrow$ Discriminator B $\rightarrow$ Real/Fake

By combining the two flows above, we get:

- Domain B $\rightarrow$ Generator A $\rightarrow$ Discriminator A $\rightarrow$ Real/Fake
- Domain A $\rightarrow$ Generator B $\rightarrow$ Discriminator B $\rightarrow$ Real/Fake

The striking feature is that the Generator models are not only create new images in target domain but also translate more reconstructed versions of input images from source domain. This can be achieved via using the generated images as input to the corresponding generator model and comparing the output image to original images. Feeding an image through both generators that is called a cycle. With this cycle consistency, each of Generator model would generate better image from source image:

- Domain B $\rightarrow$ Generator A $\rightarrow$ Domain A $\rightarrow$ Generator B $\rightarrow$ Domain B
- Domain A $\rightarrow$ Generator B $\rightarrow$ Domain B $\rightarrow$ Generator A $\rightarrow$ Domain A

Last but not least, the further element key of the architecture is the identity mapping. This means the Generator is provided with images from target domain and expected to generate the same image without changes. This addition is optional, if we add it, the results would be better when matching color of input and output images.

- Domain A $\rightarrow$ Generator A $\rightarrow$ Domain A
- Domain B $\rightarrow$ Generator B $\rightarrow$ Domain B

Loss Function. The two GAN Losses are given by:

$$L_{GAN(A2B)} = E_{b \sim P_{data}(b)}[\log D_b(b)] + E_{a \sim P_{data}(a)}[\log(1 - D_b(G_{ab}((a))] \tag{2.10}$$

$$L_{GAN(B2A)} = E_{a \sim P_{data}(a)}[\log D_a(a)] + E_{b \sim P_{data}(b)}[\log(1 - D_b(G_{ab}((a))] \tag{2.11}$$

Thanks to $L_{GAN(A2B)}$ and $L_{GAN(B2A)}$ in Equation 2.10 and 2.11, model could learn to distinguish the real and the fake data. D_a, D_b are discriminators for domain A and B, respectively. G_{ab} and G_{ba} work as generators to translate an image from domain A to B and vice versa.

However, if there is only adversarial loss in whole network, the model can only learn to map identically distributed domain. With large enough capacity, a network can map the same set of images to any random permutation of images in destination domain. Hence the adversarial losses alone can not ensure that the learned function can map an individual image in source domain to the target domain. To address this problem, cycle-consistency was invented. Cycle-consistency loss is illustrated below:

$$L_{cyc} = E_{a \sim P_{data}(a)}[||G_{ba}(G_{ab}) - a||_2] + E_{b \sim P_{data}(b)}[||G_{ab}(G_{ba}) - b||_2] \tag{2.12}$$

With the assistance of cycle-consistency loss, the network can guarantee that, for each image x from domain X, the image translation stream should be able to bring x back to the original image. Full objective is combined these losses, λ is hyper-parameter, usually set to 10.:

$$L = L_{GAN(A2B)} + L_{GAN(B2A)} + \lambda L_{cyc} \tag{2.13}$$

2.5 Image Domain Adaptation

Domain adaptation is a field associated with machine learning and transfer learning. Normally, we perform the training and testing processes on the same data distribution. However, there exists many cases that we face the problem of difference domains among trainset and testset. Domain adaptation method is leveraged to solve such problem. In details, the method tries to minimize the differences among training and target domains. Domain adaptation has also been demonstrated to be beneficial for learning among unrelated distributions. In our case, we have annotated daytime cityscapes images for semantic image segmentation purpose, meanwhile, our target is to segment on nighttime images. Thus, domain adaptation method helps in this case to adapt available daytime images to required nighttime images. Previous work have been done with some proposed methods that acquire achievements, namely [43, 38, 8, 6]. These work focus on combining models to address model adaptation. Some data augmentation techniques such as random cropping, random rotation and flipping are leveraged to stably adapt in unrelated domains [49]. There are researches studying the effective use of synthetic data [40, 48]. Pre-processing input images is also used to prevent performance degradation [36].

In [8], the authors leverage the similarity of three domains: daytime, twilight and nighttime domain. Starting with the assumption that the gap between daytime and nighttime domain could be easily bridged via multiple stages of transferring knowledge from daytime to twilight and another transferring knowledge from twilight to nighttime. On the contrary, in [43, 38], they make use of a recent striking method - GAN to distill information from one domain to another domain. To be specific, this work first uses GAN to translate data from the domain of daytime to nighttime. Then there are two models which are responsible for semantic segmentation daytime and nighttime images separately.

In this work, our main task is taking daytime cityscapes images as available inputs while our target is to semantic segmentation on nighttime cityscapes images. Insprired by ideas of mentioned work, we make use of GAN to perform a image domain translation - from daytime to nighttime. Thus, we can also leverage the annotated daytime cityscapes images (Cityscapes [7]). The target is to help our segmentation model have knowledge of nighttime image segmentation.

Chapter 3

Methodology

3.1 Overview Proposed Framework

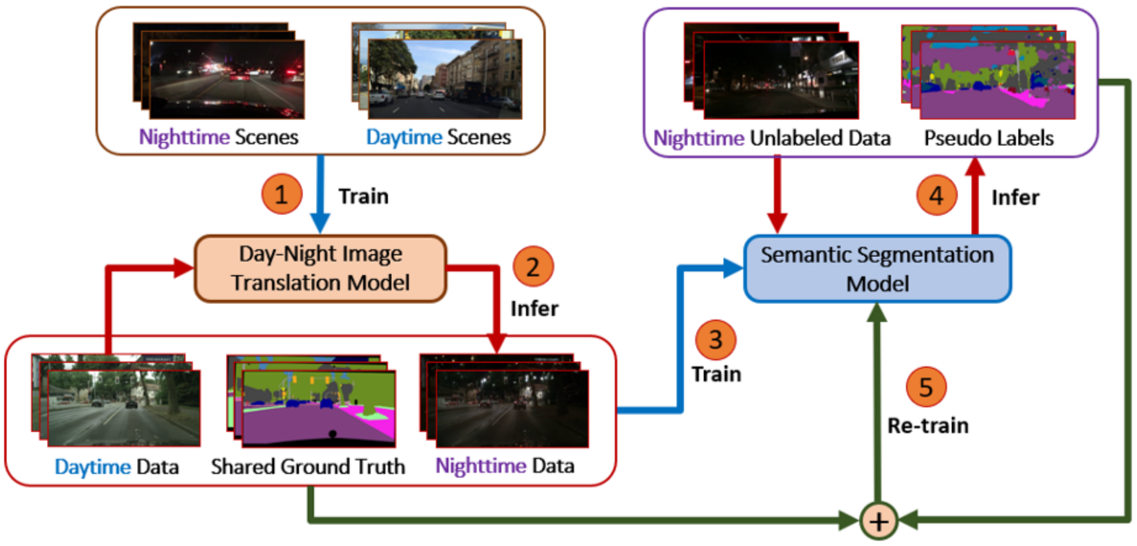

Figure 3.1: Our Semantic Segmentation with Domain Adaptation Method Framework.

In this work, we target at performing semantic image segmentation on nighttime cityscapes images with only the given daytime cityscapes dataset. Thus, we proposed to use a domain adaptation method based on GAN to cope with the problem of day and night. In the segmentation component, we leveraged self-training method to improve the system performance. Our proposed framework is illustrated in Figure 3.1.

Our proposed framework is a combination of two main components: Day-Night Image

Translation Module and Semantic Segmentation Module. The first component takes the responsibility of converting daytime cityscapes images to nighttime domain with an aim at preparing to train our segmentation model. In the next semantic segmentation component, we train our model to make prediction on nighttime images. Furthermore, we use self-training technique in this module to refine our segmentation model. To be specific, our framework consists of five main steps. *Firstly*, two sets of daytime and nighttime cityscapes images are taken to train our GAN-based image translation model. *Secondly*, the trained model translates daytime cityscapes images into nighttime domain. Both daytime and nighttime images share the same ground truth and become our segmentation training dataset. *Thirdly*, our semantic segmentation model is trained on the newborn dataset. *Fourthly*, a set of extra unlabeled nighttime cityscapes images is the input of the trained segmentation model to generate pseudo-labels. *Finally*, the combination of the inferred images with pseudo-labels and the previous training data are utilized to perform self-training stage, which we denotes as *re-train* our model. In the next sections, we present the work of the two components in our system and their specifications.

3.2 GAN-based Image Translation Component

Many computer vision problems can be seem as an image-to-image translation problem, which illustrates how to map an image in first domain corresponding to image in second domain. There are two way to deal with this problem: supervised and unsupervised learning. Supervised learning relies on pairs of corresponding images in two different domains, the opposite is true for unsupervised learning. There is no need pairs of two domain images, we only have two independent sets of images which consists of images in two different domains separately. Inspired by idea of Generative Adversarial Networks (GAN), this GAN-based method aims at converting two domain distributions: daytime and nighttime.

In this thesis, we use the previously mentioned UNIT [30] framework as the translation converter to do our task. Unsupervised image-to-image translation (UNIT for short) is based on Generative Adversarial Networks (GAN) and Variational Autoencoders (VAEs). Each image domain is modeled by the VAEs. The adversarial training objective interacts with a weight-sharing constraint, which enforces a shared latent space to generate corresponding images in two different domains. The VAEs is relevant to translated images with input images in the respective domains.

3.2.1 Variational Autoencoders - GAN

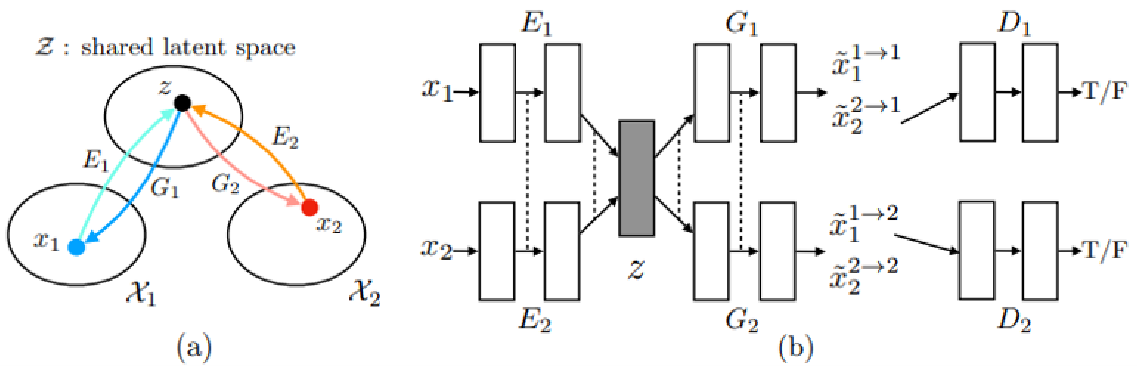

Figure 3.2: The shared latent space assumption [30].

UNIT makes the shared latent space assumption that any given pair of images x_1 and x_2, there exists a shared latent code z in a shared latent space, such that we can recover both images from this code z, and we also can compute this code from each of two images. UNIT can reconstruct the input image from translating back the translated image.

UNIT assumes a pair of corresponding images (x_1, x_2) in two different domains X_1 and X_2 can be matched to the same latent code z in a shared latent space Z. E_1 and E_2 are two encoding functions, mapping images to latent codes. G_1 and G_2 are two generation functions, mapping latent codes to images. E_1, E_2, G_1 and G_2 are implemented via CNN and satisfied the condition, shared latent space assumption using a weight constraint. As we can see, the dashed line denote for the relation weight of last few layers between E_1 and E_2, G_1 and G_2 as well. $x^{1\to1}$ and $x^{2\to2}$ are self-reconstructed images, and $x^{1\to2}$ and $x^{2\to1}$ are domain-translated images. D_1 and D_2 are discriminator for respective domain to evaluate if the translated images are fake or real.

Variational Autoencoders. The pair of encoder-generator E_1, G_1 represents a VAE for X_1 domain, called VAE_1. Take image x as an input image for instance, the VAE_1 will map x to a code in latent space Z using encoder E_1 and randomly generates a version of encoded code to reconstruct the input image via generator G_1. The components in the latent space Z are conditionally independent and Gaussian with unit variance. The output of encoder is a mean vector $E_{u,1}(x1)$ and the distribution of latent code z_1 is given by $q_1(z_1|x_1) \equiv \mathcal{N}(z_1|E_{u,1}(x_1), I)$, where I is an identity matrix. The reconstructed image is $x_1^{1\to1} = G_1(z_1 \sim q_1(z_1|x_1))$. They treated the distribution of $q_1(z_1|x_1)$ as a random vector of $N(E_{u,1}(x_1), I)$ and sampled from it. VAE_2 is similar with VAE_1 E_2, G_2 constitutes a VAE for X_2.

Weight-sharing. Weight-sharing is an idea for enforcing constraint to relate the two VAEs in framework. Particularly, the last few layers of E_1 and E_2 are shared each other responsibly for extracting high-level features of input images in two different domains. Similarity, UNIT also share the weights of first few layers of G_1 and G_2 responsible for decoding high-level representations for recovering the input images.

The weight-sharing constraint alone does not ensure that corresponding images in two different domains will have the same latent code in latent space. Because there is no pair of corresponding images in the two domains were collected to train the network to give the output at the same latent code, the latent codes which are extracted for a pair of corresponding images are different in general. Even if they are the same, the same latent component may have different meanings in different domains. Thus, the same latent code could still be decoded to generate two dissimilar images. However, By adversarial training, a pair of corresponding images in two domains can be matched to a general latent space by E_1 and E_2 and latent code will be decoded to pair of corresponding images in the two domains by G_1 and G_2.

To summary, the shared-latent space is applied to allow to perform image-to-image translation, which contains two information processing streams: $X^1 \to X^2$ and $X^2 \to X^1$. This two streams are jointly trained with the two image reconstruction streams from VAEs. Specifically, the composition of E_1 and G_2 functions approximates $I^{1\to 2}$ and the composition of E_2 and G_1 functions approximates $I^{2\to 1}$.

GAN. Like CycleGAN this UNIT framework has two GANs: GAN_1, GAN_2 defined by D_1, G_1 and D_2, G_2, respectively. With GAN_1, if real images from first domain are fed into, D_1 output true, in contrast, for images generated by G_1, it should output false. G_1 could generate two types of images involving images from reconstruction flow $x^{1\to 1}$ and translated images $x^{2\to 1}$. As we can see, the reconstruction stream can be supervised trained, so we only train to generate image from translated domain stream $x^{2\to 1}$ via adversarial training method. The similar processing to GAN_2 are applied, where D_2 is trained to distinguish true output for real sample images from the second domain and false for generated images from G_2.

Cycle Consistency. Regarding to the shared latent space assumption, which is conditional cycle consistency, we could enforce it in translation framework to regularize the unsupervised image-to-image translation problem. As a result, this information are reconstructed like a cycle stream.

3.2.2 Loss function

In order to minimize a variational upper bound, VAE loss is define below:

$$\mathcal{L}_{VAE_1}(E_1, G_1) = \lambda_1 \mathbf{KL}(q_1, (z_1|x_1)||p_n(z)) - \lambda_2 E_{z_1 \sim q_1(x_1|x_1)}[\log p_{G_1}(x_1|z_1)] \tag{3.1}$$

$$\mathcal{L}_{VAE_2}(E_2, G_2) = \lambda_1 \mathbf{KL}(q_2, (z_2|x_2)||p_n(z)) - \lambda_2 E_{z_2 \sim q_2(x_2|x_2)}[\log p_{G_2}(x_2|z_2)] \tag{3.2}$$

In 3.1 and 3.2, λ_1 and λ_2 are hyper-parameters which control the weight of objective terms and the KL divergence terms penalize deviation of distribution of the latent code from the prior distribution. The prior distribution is a zero mean Gaussian $p_n(z) = \mathcal{N}(z|0, I)$.

The GAN objective functions are given by:

$$\mathcal{L}_{GAN_1}(E_2, G_1, D_1) = \lambda_0 E_{x_1 \sim P_{x_1}}[\log D_1(x_1)] + \lambda_0 E_{z_2 \sim q_2(z_2|x_2)} \log(1 - D_1(G_1(z_2)))] \tag{3.3}$$

$$\mathcal{L}_{GAN_2}(E_1, G_2, D_2) = \lambda_0 E_{x_2 \sim P_{x_2}}[\log D_2(x_2)] + \lambda_0 E_{z_1 \sim q_1(z_1|x_1)} \log(1 - D_2(G_2(z_1)))] \tag{3.4}$$

In 3.3 and 3.4, there are conditional GAN losses which are able to translate images in source domain to target domain. λ_0 is the hyper-parameter that control the influence of the GAN.

Cycle-consistency is likely to be the same as VAE objective functions, as shown in Equation 3.5 below:

$$\begin{aligned}\mathcal{L}_{CC_1}(E_1, G_1, E_2, G_2) = \lambda_3 \mathbf{KL}(q_1, (z_1|x_1)||p_n(z)) + \lambda_3 \mathbf{KL}(q_2(z_2|x_1^{1\to 2}))||p_n(z)) \\ - \lambda_4(E_{z_2 \sim q2(z_2|x_1^{1\to 2})}[\log p_{G_1}(x_1|z_2))]\end{aligned} \tag{3.5}$$

$$\begin{aligned}\mathcal{L}_{CC_2}(E_2, G_2, E_1, G_1) = \lambda_3 \mathbf{KL}(q_2, (z_2|x_2)||p_n(z)) + \lambda_3 \mathbf{KL}(q_1(z_1|x_2^{2\to 1}))||p_n(z)) \\ - \lambda_4(E_{z_1 \sim q1(z_1|x_2^{2\to 1})}[\log p_{G_2}(x_2|z_1))]\end{aligned} \tag{3.6}$$

In 3.5 and 3.6, the KL terms penalize the latent codes deviating from the prior distribution in the cycle-reconstruction stream that leads to there are two KL terms. In addition, the negative log-likelihood objective term are able to ensure translated image resembles the input one.

As a result, UNIT deals with problems that optimizes $VAE_1, VAE_2, GAN_1, GAN_2$ to reconstruct images, the translation streams and the cycle-reconstruction streams:

$$\min_{E_1,E_2,G_1,G_2} \max_{D_1,D_2} \mathcal{L}_{VAE_1}(E_1,G_1) + \mathcal{L}_{GAN_1}(E_2,G_1,D_1) + \mathcal{L}_{CC1}(E_1,G_1,E_2,G_2)$$

$$\min_{E_2,E_1,G_2,G_1} \max_{D_2,D_1} \mathcal{L}_{VAE_2}(E_2,G_2) + \mathcal{L}_{GAN_2}(E_1,G_2,D_2) + \mathcal{L}_{CC2}(E_2,G_2,E_1,G_1)$$

3.2.3 Training

Due to the numerous hyper-parameters of training process, we randomly crop image with a patch of 256×256 to train the UNIT. By this way, we can only infer images with the size of 512×1024 which is below 1024×2048 the original resolution of images in NEXET. Therefore, before feeding into segmentation model, we have to upsample the inferred images to 1024×2048 although this operation makes inevitable influence on the final result. We first train the UNIT model on our customized dataset with the default hyper-parameters that the author mentioned. $\lambda_0 = 10, \lambda_1 = \lambda_3 = 0.1, \lambda_2 = \lambda_4 = 100$.

Configuration. Configuration of training process is set by default in the first training (for people who are interested, please refer [30] to read for more information) , then we modified some hyper-parameter suitable for our dataset. Particularly, image size from 256 to 512, adding layer normalization, adding perceptual loss.

3.2.4 Perceptual loss maintains the semantic features

As not available of day-to-night dataset, we are facing to this problem to perform the translation image-to-image task. So we decided to crawl some dataset that have the same distribution with day and night domains. As a result, 50000 images are collected with day, night and twilight conditions. Separating by histogram, we finally obtained 19858, 19523 images in daytime and nighttime domain, respectively. We first trained the UNIT model on our customized dataset. Some problems have been found, especially a vast shiny sparkling point such as vehicle light, traffic light as shown in Figure 3.3. To address these problems, we added the perceptual loss to diminish the failures.

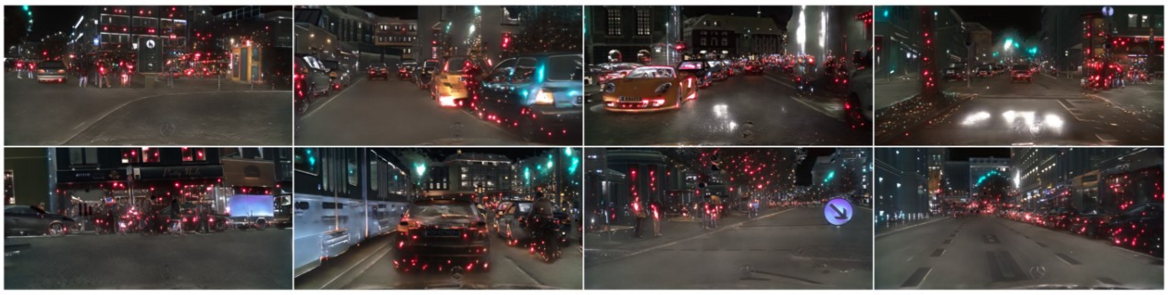

Figure 3.3: Image-to-Image translation results in preliminary stage.

In this thesis, changing the loss function of training process plays a crucial role in all GAN framework, specifically, adding perceptual loss helps the UNIT model to generate results that look more convincing and realistic. In the training process, we added the perceptual loss with the weight $\lambda_{vgg} = 1$ and the result is given by 3.4.

To explain it, there are several reasons. Firstly, the result before adding perceptual loss shows that there are various mismatched colors with traffic lights or vehicle light that are a result of training dataset. Secondly, in this case, because of the purposes of perceptual loss, it would be able to deal with problem above. As the author mentioned, the perceptual loss helps for some datasets, particularly for synthetic to real. The perceptual loss aims at mapping the features between two images instead of only comparing the value of each pixel in the image.

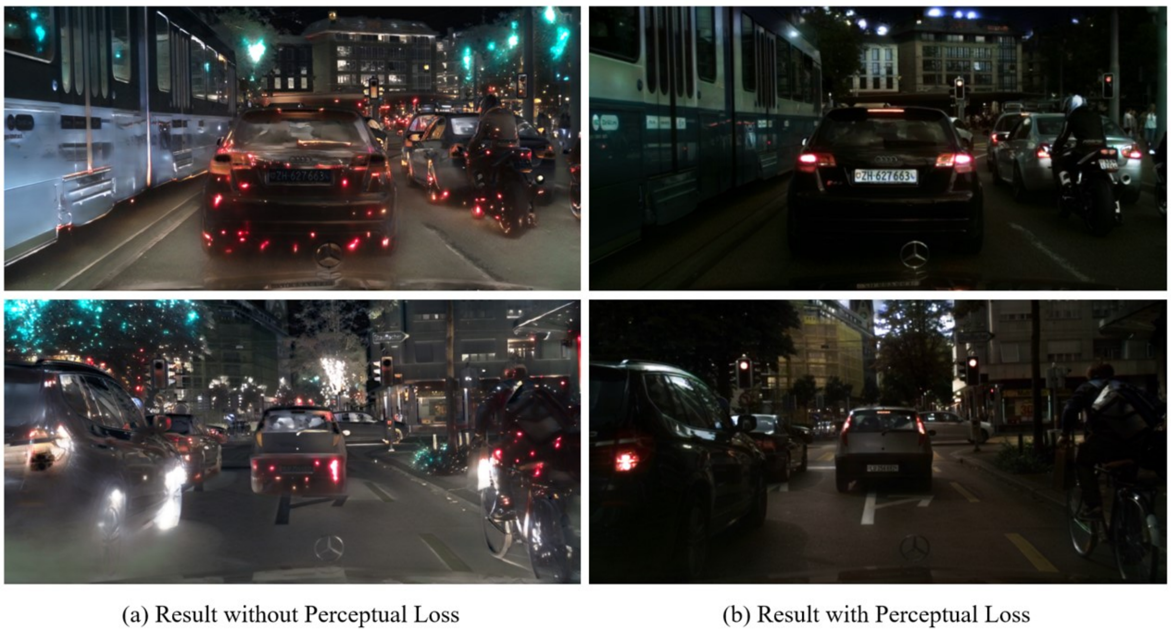
(a) Result without Perceptual Loss
(b) Result with Perceptual Loss

Figure 3.4: Comparison of the result when the Perceptual Loss is applied and not.

The figure 3.4 shows that the comparison between the result applied the perceptual loss and no. (a) The results without the perceptual loss seem to have many sparkling points that cause the entire image does not looks realistic. (b) The results applied the perceptual loss reduce the wrong light matching point. In this case, the perceptual loss uses VGG pretrained model to extract the features of images especially light of vehicle and traffic. Then, these features are compared between two domains daytime and nighttime. The higher perceptual loss means that there is a large amount of mismatch features particularly sparking points, so we minimized the perceptual loss to tackle this problem of sparkling points.

3.2.5 Perceptual Loss

Some recent researches show that perceptual loss in Figure 3.5 (feature loss or perceptual loss) plays an vital role in assessment of features of images. VGG is Convolutional Neural Network (CNN) backbone which proposed in [18] in 2014, the perceptual loss is based on VGG to evaluate the generator model via the extracted features. Particularly, when an image is fed into the VGG model, various detected features are calculated to measure the perceptual loss. The perceptual loss contains feature losses and style loss, for example, a feature map has 256 channels with 28×28 width and height, which are used to determine features such as eyes, mouth, lip, face and so on. The output at the same layer for the original image and the generated image are compared via mean squared error (L_2) or the least absolute error (L_1). As a result, the model could produce much finer detailed images that is function of features losses.

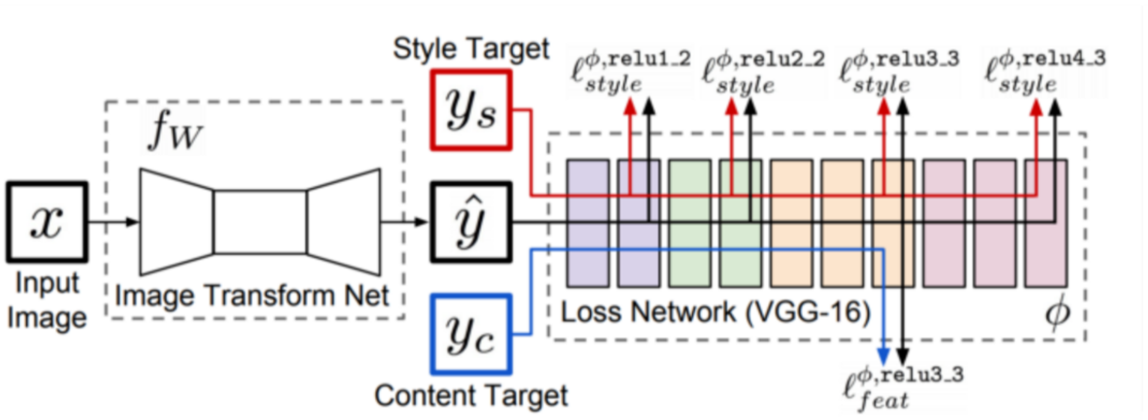
Style Target
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
f_W
y_s
x
$\hat{y}$
Input Image
Image Transform Net
y_c
Loss Network (VGG-16)
ϕ
Content Target
$\ell_{feat}^{\phi,relu3\_3}$

Figure 3.5: Overview of perceptual loss. Loss Network (VGG-16) is pretrained for image classification to calculate the perceptual loss that evaluate the differences in content and style between images [18].

3.3 Semantic Image Segmenation with Self-training Strategy

After the work of the previous image translation component to translate cityscapes images in daytime to nighttime condition. In this module of semantic image segmentation, we leverage the Panoptic Feature Pyramid Networks (Panoptic FPN) [20] as our backbone model to perform the semantic segmentation task. To the aspect of training strategy, we apply self-training technique [55] rather than using pre-trained or training our model from scratch. Details are explained in the section below.

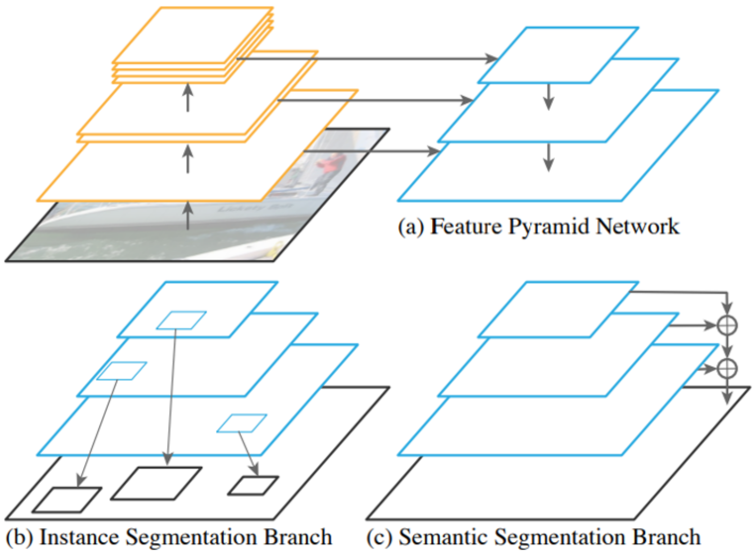
(a) Feature Pyramid Network
(b) Instance Segmentation Branch
(c) Semantic Segmentation Branch

Figure 3.6: General Panoptic Feature Pyramid Networks Architecture [20].

3.3.1 Panoptic Feature Pyramid Networks

Panoptic FPN is similar to a so-called popular Mask R-CNN [13] approach with a shared Feature Pyramid Network [27] (FPN) backbone. As mentioned by the authors, this Panoptic FPN yields a lightweight model and a top performance method for both semantic and instance segmentation.

FPN-based Networks. The very first FPN was published for the task of object detection [27] in the year of 2017. FPN extracts information at multiple spatial resolutions (with the help of ResNet [14] for instance) and adds lateral connections, as in Figure 3.6a. The top-down pathway starts from the deepest layers of the network and those are upsampled to the larger size. FPN generates a pyramid, typically with the resolution scales from 1/32 to 1/4, where each pyramid level has the same channel dimension of 256 by default. We made use of such concept of this network to do the task of instance segmentation and semantic segmentation (as in Figure 3.6b,c).

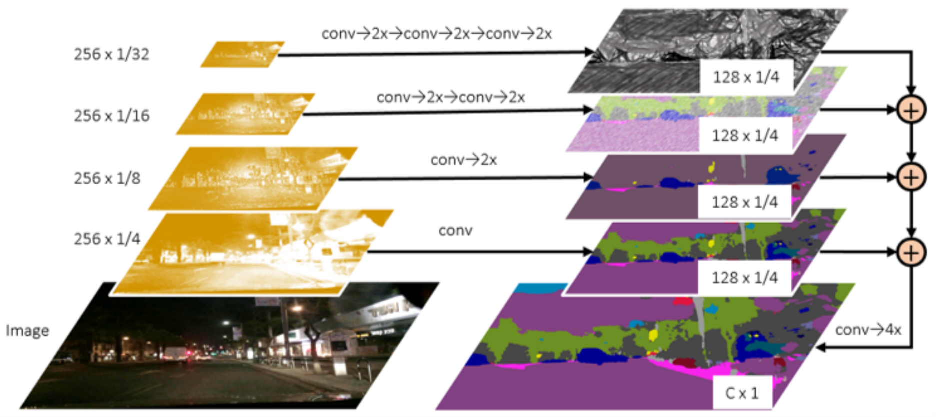

Figure 3.7: Panoptic FPN Architecture for Semantic Segmentation.

Panoptic FPN. Panoptic FPN is divided into two parts: encoder and decoder. In the encoder stage, Panoptic FPN extracts features of the image in multi-scale, which allows the network to capture better spatial information of objects in the image. There are four levels of feature map in the pyramid (at 1/32, 1/16, 1/8 and 1/4 scales) and each contains different levels of semantic information. Then in the decoder stage, features maps of multi-scale are used to recover the required output. To generate semantic segmentation output from FPN features, there is a simple design in this network (as in Figure 3.7) whose purpose is to merge the information from all levels of the pyramid network into a single output (segmentation map). The three feature maps with scales of 1/32, 1/16 and 1/8 are upsampled to the size of the 1/4 scale. Each upsampling process consists of 3×3 convolution, group norm [46], ReLU, and $2\times$ bilinear upsampling. The result output would be the element-wise sum of the four feature maps. Then a combo of final 1×1 convolution, $4\times$ bilinear upsampling, and softmax are used to generate the per-pixel class labels at the original image resolution.

Discussion. Panoptic FPN is able to capture fine structures of images thanks to the multiple resolutions encoder stage. Moreover, the encoder can extract sufficiently rich semantic information at each resolution level to predict class labels. Based on FPN, Panoptic FPN with some refinement of the network can adapt well the segmentation tasks.

3.3.2 Proposed Loss Function

In this task of semantic image segmentation, we used two typical types of loss functions which are Cross Entropy Loss and Focal Loss. In our framework, we also proposed a loss function that combine the specifications of the two mentioned losses. This section presents striking features of the two existing loss functions together with our proposed one.

Cross Entropy Loss. Cross entropy loss (a.k.a. log loss) targets at measuring the performance of a classification model whose output is a probability value between 0 and 1. To the problem of binary classification, the considered model gives output of whether or not the input image belongs to a specific class. Cross entropy loss is used to measure this difference and thus, optimizing this loss function also minimizes the difference of the predicted label compared with ground truth annotation.

Despite the fact that cross entropy loss is known to apply for the task of image classification, we can also leverage this feature to apply it to other task. Particularly, in the task of semantic image segmentation, we assume that the segmentation model has to assign each pixel a semantic label among given labels of ground truth. Therefore, it is considerable that semantic image segmentation is a task of classification - pixel label classification. Then, we can apply cross entropy loss to measure the differences among classes of objects in the images.

Cross entropy can be calculated as the Equation 3.7, where y is the label and $p(x)$ is the predicted probability of the inferred label for all N samples:

$$\mathcal{C}ross\_Entropy(p(x_i), y_i) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \cdot log(p(x_i)) + (1 - y_i) \cdot log(1 - p(x_i))) \quad (3.7)$$

We can also consider this loss function simply as this Equation 3.8, in which p is the predicted probability and y is the label with value 1 or 0:

$$\mathcal{C}ross\_Entropy(p(x), y) = \begin{cases} -log(p(x)) & \text{if y=1} \\ -log(1 - p(x)) & \text{otherwise.} \end{cases} \quad (3.8)$$

Focal Loss. To our knowledge, cross entropy is a widely used loss in classification problems. And in the task of segmentation, we leverage it for each pixel classification task. However, there exists problem of this loss function in some specific cases. For instance, small object areas in the image would be overwhelmed by large ones. Normally, when forwarding an image through a segmentation network, after which a sigmoid function is used to convert the prediction to a probability value. Then, binary cross entropy loss is used to perform back-propagation. However, as the loss is back propagated as a whole, it will be difficult for the model to learn the labels of small objects. In general, what we need to solve is to help the model predict semantic labels effectively even though the label distribution is highly unbalanced. Focal loss is introduced to face with such problem.

Focal loss is designed to cope with highly imbalanced datasets. This is an improved version of cross entropy loss, which is known as a **more focused cross entropy loss**. This loss function tries to decrease the total loss based on the feedback of each pixel. By decreasing the total loss, the chance of overwhelming is also decreased. By referring to the feedback accuracy of each pixel during training phase, the losses of well-trained pixels are largely decreased, while the losses of poorly-trained pixels are only **trivially decreased**. Focal loss [26] is an implementation of this idea.

Let's unify the two equations of Equation 3.8 into one with the replacement of p_t instead of p and $1-p$ in the Equation 3.9, the result is the following Equation 3.10:

$$p_t = \begin{cases} p(x_i) & \text{if y=1} \\ 1-p(x_i) & \text{otherwise.} \end{cases} \tag{3.9}$$

$$\mathcal{C}ross\_Entropy(p, y) = Cross\_Entropy(p_t) = -log(p_t) \tag{3.10}$$

Particularly, if label $y = 1$, then $p_t = p$ and a bigger p_t presents a bigger p, which is closer to label 1. If label $y = 0$, then $p_t = 1 - p$ and a bigger p_t presents a smaller p, which is closer to label 0. As a result, p_t yields the accuracy of the prediction of our model: the bigger p_t, the better model performance. Nevertheless, our objective is reducing the loss of pixel if its prediction is good. Then, the author of the loss leverages p_t to minimize the loss function as p_t is the measurement of prediction accuracy. Again, bellow is the form of focal loss introduced by Lin et al [26]:

$$\mathcal{F}ocal\_Loss(p_t) = -\alpha_t(1 - p_t)^{\gamma} \cdot log(p_t) \tag{3.11}$$

In details, focal loss is just a refined version of cross entropy loss by adding a weight of $-\alpha_t(1 - p_t)^{\gamma}$. Here, $1 - p_t$ is used as a factor to decrease the original cross entropy loss with the help of two new hyperparameters: α_t and γ. As mentioned above, if p_t

gets bigger and is closer to 1, $1 - p_t$ gets smaller and is closer to 0, thus the original cross entropy loss is largely decreased. On the contrary, if p_t gets smaller and is closer to 0, $1 - p_t$ gets larger and is closer to 1, thus the original cross entropy loss is **trivially decreased**. Experimental results of hyperparameters in the original paper of focal loss [26] show that the hyperparameter α_t is in range of 0 and 1, and the good γ is set to 2.

Our Proposed Combined Loss Function Either cross entropy loss or focal loss maintains good specifications that benefit our segmentation model. From that point, we proposed a combined loss function that is a combination of the two mentioned functions. Cross entropy loss measures the differences between each couple of pixels in the ground truth and the predicted mask regardless of whether they are the major or the minor objects in the image. Meanwhile, focal loss focuses on balancing the importance of small area objects to be equally treated as large area ones. We denote L_{pixel} as cross entropy loss function since this loss measures the differences among couple of pixels. Similarly, we denote $L_{balanced}$ as focal loss since this loss is able to balance the importance of minor and major pixels in the image. Equation 3.12 is our proposed loss function. The effect of each component is decided by the weights of α. By default with no prior knowledge, we assume the importance of the two component is equal ($\alpha = 0.5$).

$$\mathcal{L}_{combined}(p_t) = \alpha L_{pixel}(p_t) + (1 - \alpha) L_{balanced}(p_t) \tag{3.12}$$

3.3.3 Our Self-training Method

In many tasks of machine learning or deep learning, it is undeniable that labeled data plays an important role in the success of the learning process. Let our model totally learn from labeled data is called *supervised learning*. However, there are couple of issues that arise when forming standard labeled dataset, which can be listed as: **time-consuming labelling data** or **financially expensive to collect data** and so on. Meanwhile, unlabeled data is considered as an enormous resource. Thus, this is the playground of *semi-supervised learning*. In this approach, first we train our model with labeled data to result in a assumed-fine-trained model. Then we use that model to make predictions on the unlabeled data. And for sure, those predictions of segmentation maps can be either right or wrong, but at least they are better than having no labels. Those predicted labels are called *pseudo-labels*. After an amount of training iterations, we can adopt them to continue training our model with the extra data. This specific technique is called **self-training**.

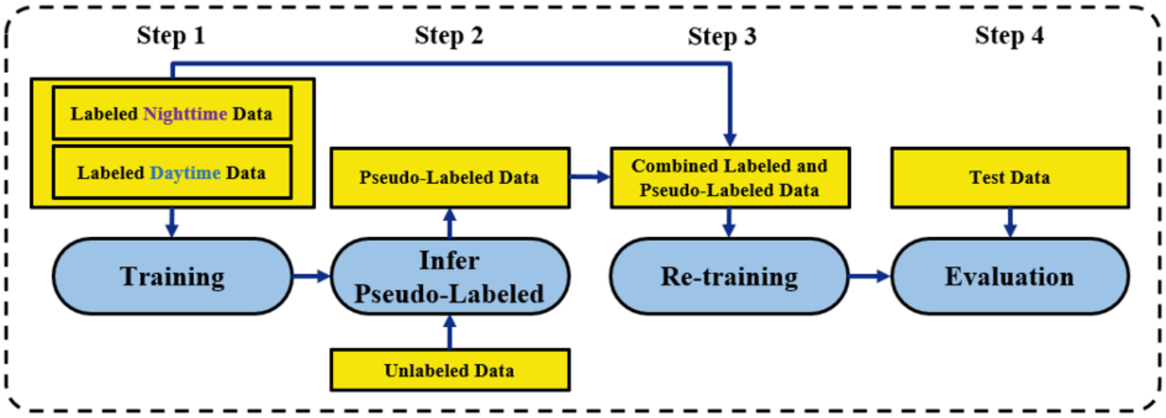

Figure 3.8: Illustration of self-training process in semantic segmentation module.

In the work of Rethinking pre-training and self-training [55] by B Zoph et al in 2020, the author proved the efficiency of self-training to the task of semantic segmentation problem to attain state-of-the-art performance. Self-training with extra unlabeled data always helps to increase the performance of the model is a statement of this publication. Dengxin Dai et al (in 2018) proposed dark model adaptation method [8] based on the idea of self-training. The method built a bridge of twilight images to narrow the differences of daytime and nighttime images and leveraged self-training to infer pseudo-labels of training images. In details, to predict segmentation on nighttime images, the model is first trained on daytime images, and infer twilight images pseudo-labels, then train the model with those pseudo-data.

In this work, we make use of the idea of the two publications above to come up with our proposed framework. First of all, we use standard daytime dataset of Cityscapes [7] as a resource for the image translation module to generate nighttime images. Together they become the day-night dataset of our work. After training the segmentation model with that amount of annotated data, we apply self-training on another unlabeled set of true nighttime images (the ratio should be a quarter of the annotated data, as our experimental results in the next chapter). Leveraging the idea of domain adaptation of Dengxin Dai et al, we use GAN to convert the domain of images from day to night instead of using twilight images, with the expectation that our model can handle nighttime cases.

Figure 3.8 depicts in conceptual level how self-training works in our semantic segmentation module. In *Step 1*, our semantic segmentation model is trained with a combination of labeled daytime cityscapes images and labeled nighttime cityscapes images. Note that the nighttime images of cityscapes were generated by our GAN-based image translation module and their shared labels were those of daytime ones. After a number of iterations training the model, we perform *Step 2* as a labels inference process. This step takes a set of unlabeled data as its input and uses the assumed-fine-trained model to generate pseudo-labels. The generated labels are assumed to be ground truth of those extra data. Then in *Step 3*, we combine the true-labeled data with the pseudo-labeled data together to re-train our model. Finally, *Step 4* denotes the testing phase.

Chapter 4

Experiments

4.1 Overview

This section presents the work of our experiments to demonstrate our hypothesis. Firstly, we introduce datasets for image translation component and semantic image segmentation. To be specific, in Day2Night image domain translation module, we crawl and select images which are suitable for our purpose. Then, some customized datasets are chosen using histogram and Frechet Inception Distance (FID) [15] to train the segmentation module with self-training method. The results of the first component are illustrated in the next section, which was followed by the enhancement technique by modifying the loss function. Finally, in semantic segmentation part, we present some evaluation metrics and the development path of our framework in details. We gain an insight into each burst step in our experiments.

4.2 Datasets

4.2.1 Datasets for Image Domain Translation

This section clearly describes dataset problems and several available ways to solve it.

Available Datasets for Day to Night Conversion. In recent years, Segmentation has exponential and significant achievement on semantic image segmentation. Especially, nighttime image semantic segmentation has been the talk of town due to its challenges such as dataset, environment conditions, illumination and so on. To deal with these matters, several datasets were manually created such as BDD100k [52], ZJU dataset [38], NEXET[1]. All of these have plenty of day and night scenes, which is satisfied for training GAN model.

BDD100k dataset includes a wide range of driving videos under various weather conditions, time, and scene types. To be more detailed, the dataset contains diverse scene types such as city streets, residential areas and highway. In addition, the videos were recorded in diverse weather conditions at different times of the day. In total, it has 100000 driving video (40 seconds per each), collected from more than 50000 rides, covering New York, San Francisco Bay Area and other regions.

Zhejiang University (ZJU) dataset were captured in Yuquan campus in Hangzhou (China) with multi-modal stereo vision sensor. One remarkable feature is that the areas near to the camera have good illumination. However, it is very dark for far areas because of a lot of shadows. It contains 6848 and 6282 images with 1920×1080 resolution in daytime and nighttime conditions, respectively.

Image-to-Image Translation Dataset. After doing some surveys about the dataset for day-to-night image translation, we already found some available datasets such as Cityscapes and NEXET. Because of the similarity of distribution between Cityscapes and NEXET, we first used NEXET to train UNIT to translate daytime images to nighttime images. These translated images are under consideration as data for segmentation model. NEXET dataset simultaneously contains 50000 images including daytime and nighttime images. Before training with data, we pre-process data by filtering images with Histogram Equalization via threshold α to split images into day and night domain. To be more specific, we divide all NEXET dataset into two sets: daytime, nighttime instead of three sets as given default: daytime, twilight and nighttime. Then the threshold α is applied to the two sets. If histogram of an image is lower than α_{night}, it seem to be nighttime image, in this case we set $\alpha_{night} = 65$. As a result, we finally obtaine 19858 and 19523 images for daytime domain and nighttime domain to train UNIT, respectively. We split this dataset into trainset and valset with the ratio $3 : 1$. So we pick out 4979 and 4644 images for daytime and nighttime valset, respectively. In addtion, 14937 and 14879 images are under consideration as two sets of day and night to train model.

[1]https://www.kaggle.com/solesensei/nexet-original

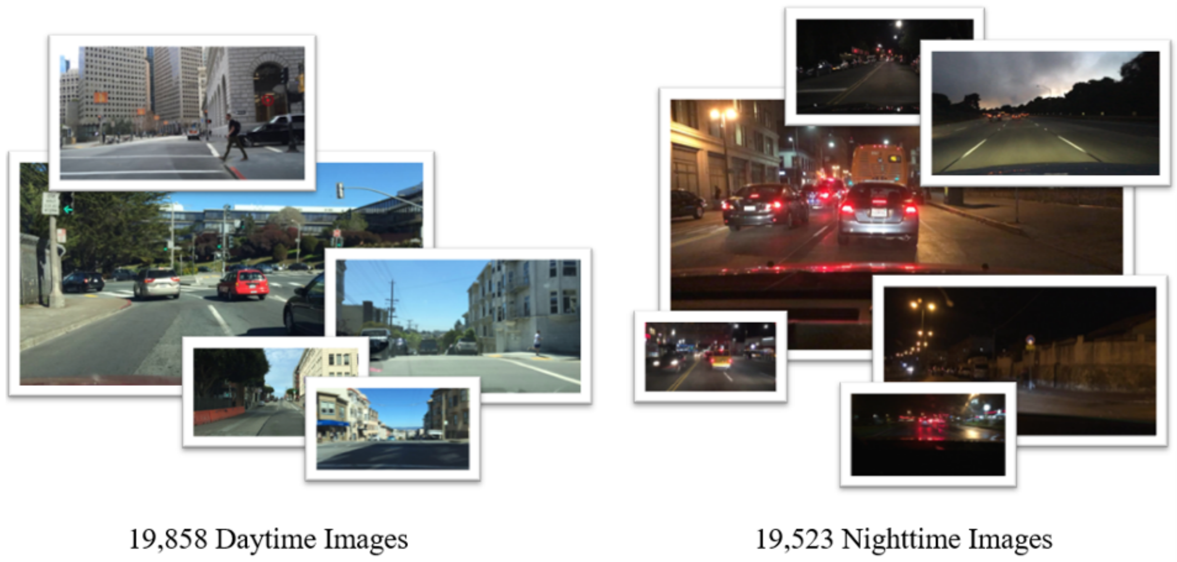

Figure 4.1: Modified dataset contains two different domains which are daytime and nighttime distribution.

4.2.2 Datasets for Semantic Image Segmentation

4.2.2.1 Cityscapes Benchmark

In this work, we proposed a framework for semantic segmentation with domain adaptation method aiming at segmenting nighttime cityscapes images. Thus, we choose to perform our experiments on Cityscapes dataset.

CityScapes[2] [7] is a dataset that contains images of many cities around the world, mostly in Europe. Images of 50 cities were captured in several months, in daytime and in fairly good weather conditions. The original Cityscapes dataset includes 30 classes as shown in Table 4.1. Marker (*) denotes objects that are not included in any evaluation

No.	Group	Objects
1	Flat	road, sidewalk, parking*, rail track*
2	Human	person, rider
3	Vehicle	car, truck, bus, on rails (train), motorcycle, bicycle, caravan*, trailer*
4	Construction	building, wall, fence, guard rail*, bridge*, tunnel*
5	Object	pole, pole group*, traffic sign, traffic light
6	Nature	vegetation, terrain
7	Sky	sky
8	Void	ground*, dynamic*, static*

Table 4.1: Cityscapes Dataset Class Definitions.

[2]https://www.cityscapes-dataset.com/

and treated as void. Thus, our used dataset includes 19 semantic classes and a void class. Figure 4.2 illustrates some examples of Cityscapes images.

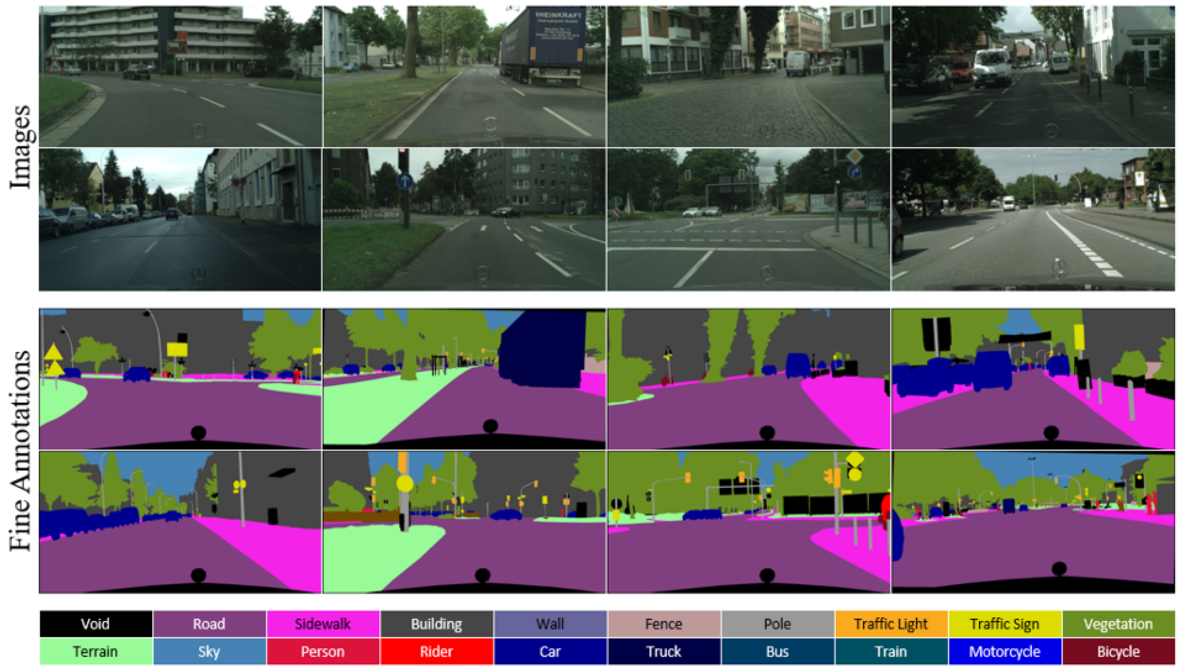

Figure 4.2: Exemplary images of Cityscapes Dataset.

The total dataset contains 25000 images divided into two volumes. A volume consists of 5000 annotated images with fine annotations. Another is 20000 annotated images with coarse annotations. Considering the system performance, we chose the fine annotations volume as our main dataset. We refer this original Cityscapes as daytime cityscapes images. Later, those images would go into our image translation module to generate nighttime cityscapes images.

4.2.2.2 Nighttime Driving Testset

As mentioned above, our framework targets at segmenting nighttime cityscapes images. Thus, we need a general testset to evaluate our system. In 2018, Dengxin Dai and Luc Van Gool published [8] with a contribution of Nighttime Driving dataset, which can be used as a benchmark for our purpose.

Nighttime Driving Test includes 50 cityscapes nighttime images. Those images are annotated for semantic segmentation purpose using 19 evaluation classes of the Cityscapes dataset, as mentioned in Table 4.1 (without * marked objects). With objects marked as *, this testset assgins them to void class. Our experiments will be evaluated on this testset. Figure 4.3 shows exemplary images along with their annotations of this testset.

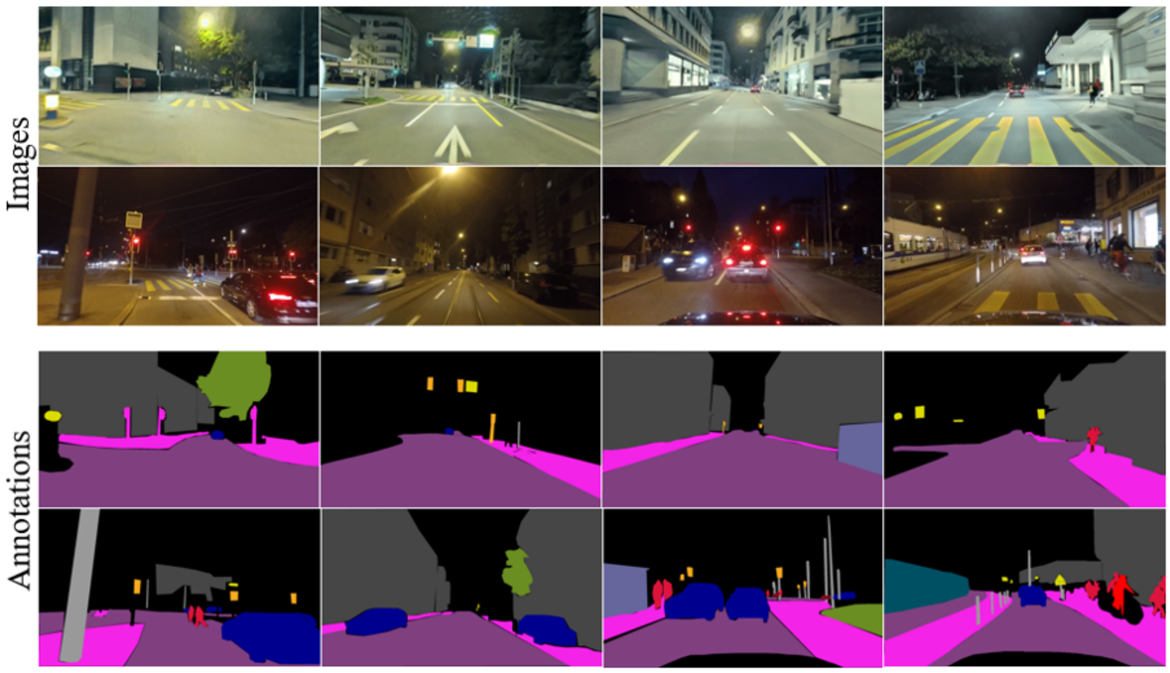

Figure 4.3: Exemplary images of Nighttime Driving Test.

4.2.2.3 Extra Unlabeled Data Selection

During the process of doing experiments, we realize that there exists a relation in the proportion of labeled images and pseudo-labeled images. When performing self-training with a dominated amount of unlabeled images compared to labeled trainset, we suffer from poor performance of models. It can be explained as the domination of unlabeled data confuses the trained model. To address this problem, we reduce the amount of unlabeled images to a suitable ratio with the help of two methods: histogram-based and FID-based.

Histogram-based Method. Firstly, we use image histogram as a filter to select images belonging to nighttime distribution. One more striking attribute is that the mentioned sparkling light. All of them seem to be dominated by red color, so we decrease the impact of red channel by a half when calculating histogram of an image shown in Figure 4.1. In conclusion, we finally choose images that have histogram between 15 and 20. Note that α in Equation 4.1 is 0.5.

$$\mathcal{S} = \alpha * R + G + B \tag{4.1}$$

FID-based Method. The FID-based method was applied to choose images. All selected image would have the same characteristic with destination domain, nighttime domain in this case. Frechet Inception Distance (FID for short) is a method to measure the distance between image and image or distribution and distribution. It is widely used for evaluated GAN model that calculates the distance of feature vector of real image and generated image. This FID score summarizes the the similarity between two sets of images in terms of statistics on extracted features using the pretrained Inceptionv3 model. So lower scores indicate that the two groups of images are more similar or have more similar distribution on statistics. The ideal score is 0 that means two sets of images are identical. In Equation 4.2, Tr sums all the diagonal elements, μ is mean, Σ is covariance. According to our experiments, we finally pick 1600 images from 14937 nighttime images with the FID threshold of 450.

$$FID(x, y) = ||\mu_x - \mu_y||_2^2 + Tr(\Sigma_x + \Sigma_y - 2(\Sigma_x \Sigma_y)^{1|2}) \tag{4.2}$$

4.3 Day2Night Image Domain Translation Component

This section shows the quantity as well as quality of experimental results of the first stage in our framework, then we discuss about strong points and weak points of this component.

4.3.1 Initial Results

In preliminary stage, we first trained GAN-based image translation method with our customized dataset. Our dataset contains 14937 daytime images and 14879 nighttime images which are in two day and night domain, respectively. This experiment shows that initial results contain some failure cases, especially wrong located traffic and vehicle lights.

As the mentioned problem of dataset above, the very first results show that the image-to-image translation task is able to translate an image from daytime domain

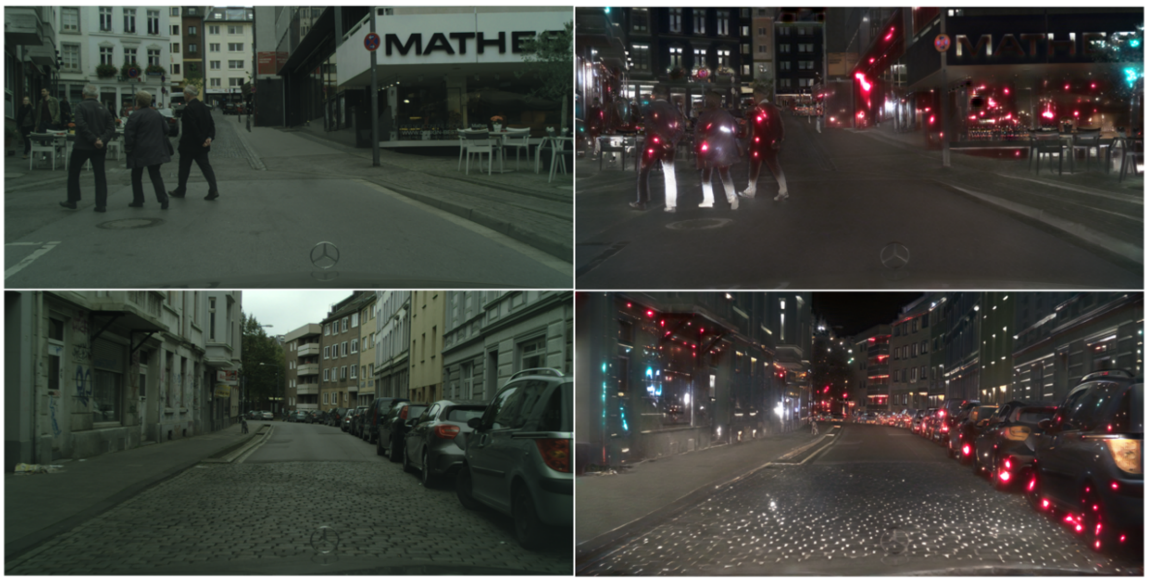

Figure 4.4: Results at 280k iterations. These images seem to be wrong located traffic and vehicle lights.

to nighttime domain without paired dataset. In Figure 4.4, we found that the results contain the features of nighttime domain, but there are a various wrong sparkling points like vehicle light and traffic light. This problem is explained in the following section.

4.3.2 With Perceptual Loss Refinement Results

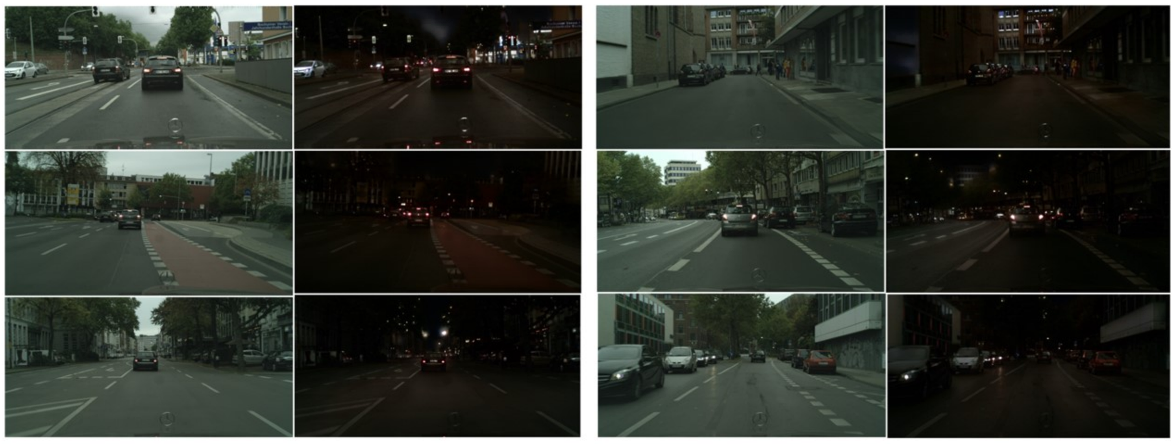

Figure 4.5: Image-to-Image translation results with additional Perceptual loss.

Fine Cases. The Figure 4.5 illustrates the results of image-to-image translation task in all segmentation framework. Almost features from daytime images are relatively converted to nighttime images. Especially, the light of vehicle or traffic looks more suitable and realistic, which is solved by adding the perceptual loss.

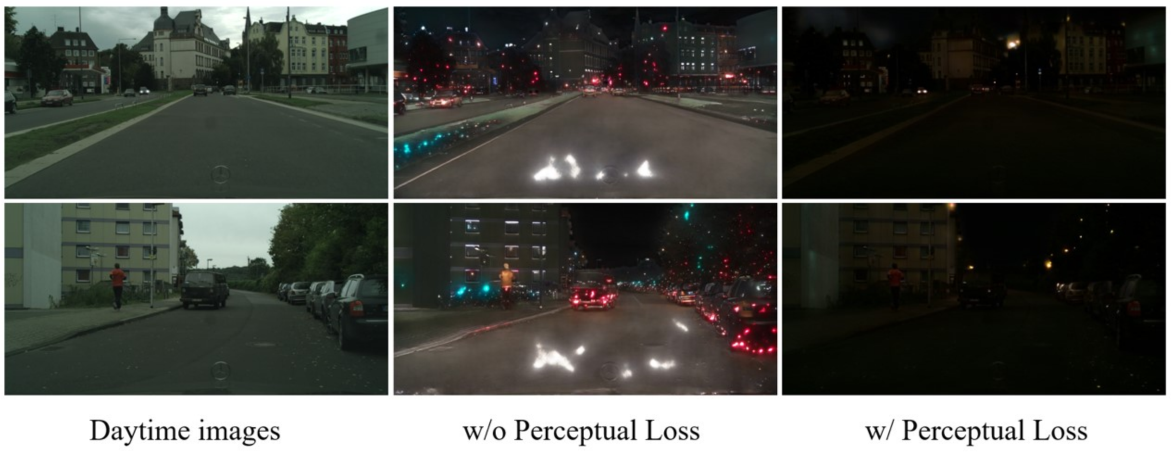

Figure 4.6: Image-to-Image translation results. It is too dark even for human vision.

Failure Cases. Although the majority of the results acceptably look convincing, some failure cases still exist such as in Figure 4.6 and 4.7. It is obviously seen that the translated images in the third column are too dark for the human vision. Translated images with the perceptual loss in Figure 4.6 are lack of illuminance, we hardly see any object without totally concentrating on the dark images. A glance at these images only give some bright points.

In contrast, not only for the dark, but also for the bright, there are so many bright images were translated by the trained model in Figure 4.7. For example, the translated images with the perceptual loss show the wrong color of sky in night condition, in this case, brightness instead of darkness sky. Moreover, the color of buildings should be also dark instead of bright. They are tough issues because of the wide range of data distributions. To explain this, we found some mismatches of dataset when grouping them into two separable daytime and nighttime domains.

Quantitative Result of Day2Night Translation Component. As mentioned FID above, we also calculate the differences between translated and real images in Table 4.2. FID_{Night} denotes the distance between translated nighttime images and real nighttime images. The similarity with daytime images is true for FID_{Day}

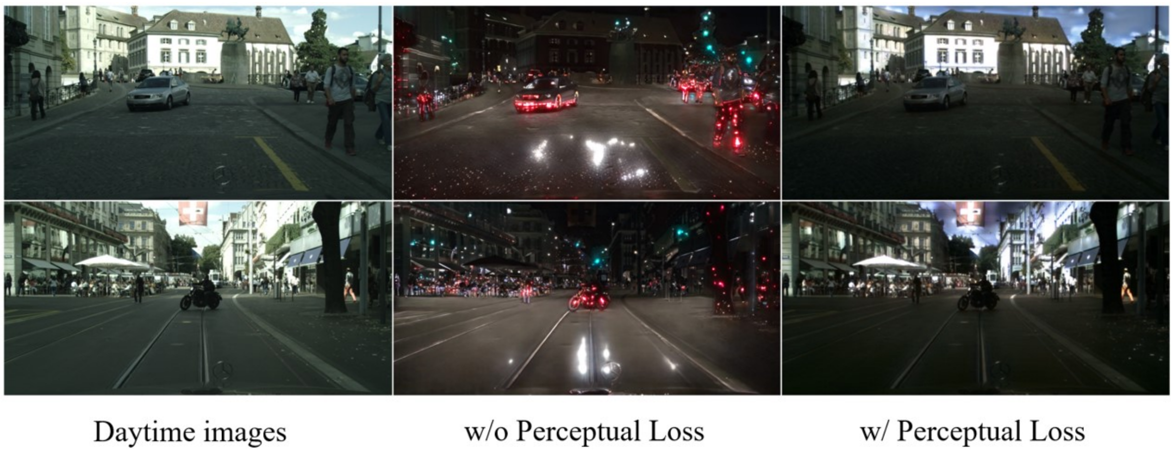

Figure 4.7: Image-to-Image translation results. The results look too bright in comparison with nighttime distribution.

ID	**Method**	**FID_Night**	**FID_Day**
1	UNIT w/o Perceptual Loss	98.39	74.45
2	UNIT w Perceptual Loss	97.68	81.92

Table 4.2: Quantitative Result of Day2Night Translation Component.

Discussion. In the first stage of the whole framework, image-to-image translation phase, we crawl data from the NEXET dataset and appropriately customize for training the UNIT model. After using default configuration for training, we recognise that the results are too bright and have a various shiny sparkling points demonstrating for wrong vehicle and traffic light. The very first results show that the translation model could perform well in its task. The daytime images are consistently transformed to nighttime images, which is the main purpose of translation method. To address the problem of too brightness and a vast shiny sparkling spots, we modify the loss function via adding the perceptual loss. The perceptual loss (VGG loss) helps to decrease the large number of brightness points in the translated images via comparing the extracted features through VGG model. We finally succeed in dealing with this problem although there are still some tough failure cases.

To the initial results without the perceptual loss, we found the explanation for some failure cases is dataset problem. When double checking the dataset, we realized that there is a wide range of images which contains a huge amount of sparkling light. Taking Figure 4.8 for instance, these pictures look too bright due to many vehicles and traffic lights. That leads the model to learning the features of vehicle light and generating them randomly in the translated images.

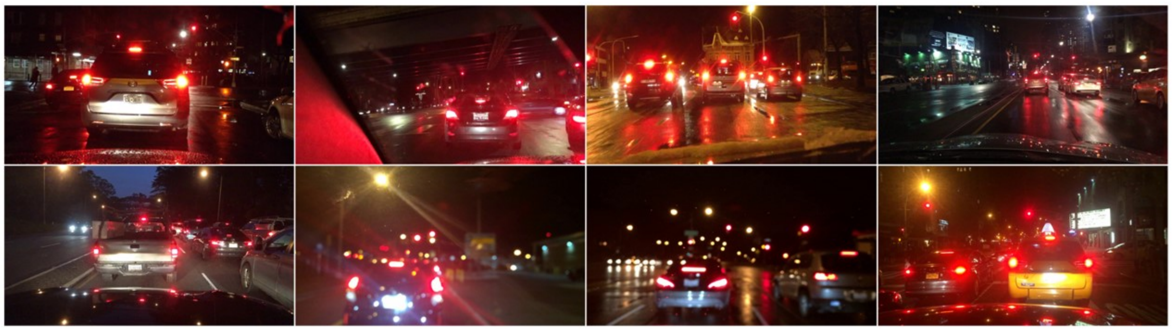
Figure 4.8: Examples of shiny sparkling of vehicle light in dataset.

When observing the failure cases, particularly extremely dark images, we also found that there is also a lot of extremely dark images in the dataset. It can be seen in Figure 4.9, these are some images taken from the nighttime domains. It is really hard to gain information from these pictures even by human vision.

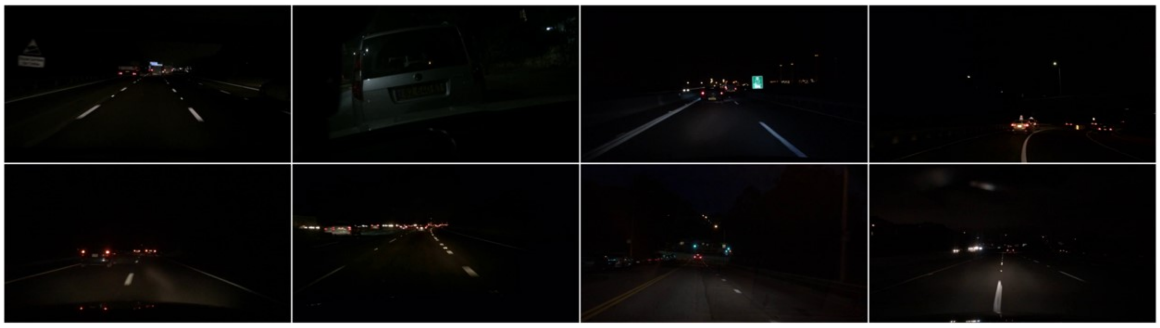
Figure 4.9: Examples of dark images in dataset.

The similarity is true for the bright images. We also found there are many bright images in the nighttime domains. In Figure 4.10, we see some examples of twilight images or even daytime images taken from the nighttime set. These images increase noise in the translation task, which lead to generating some images with brightness condition.

4.4 Semantic Image Segmentation Component

This section presents a series of experiments related to our segmentation module as well as the overall performance of our system. The segmentation module is established with self-training technique. Below we present evaluation metrics used to test our system performance and then our experimental results.

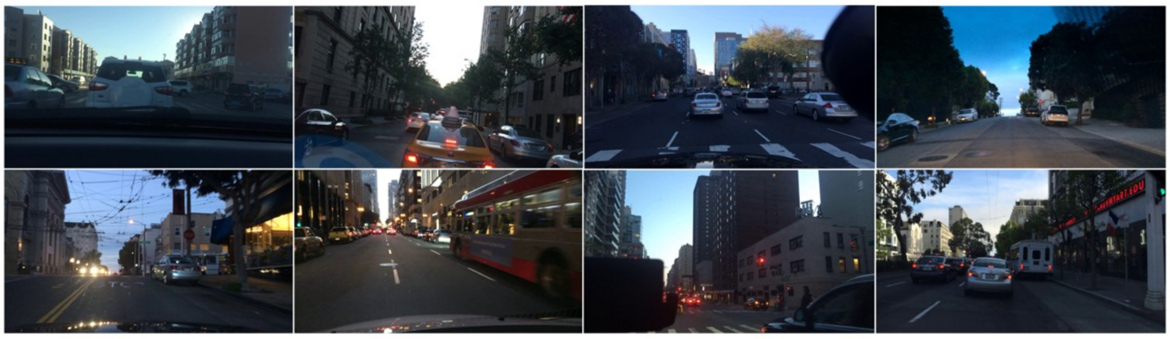

Figure 4.10: Examples of twilight images in dataset.

4.4.1 Evaluation Metrics

In this work, we evaluate our system with the two main common segmentation evaluation metrics, which are pixel accuracy and mean intersection-over-union (mIoU).

Pixel Accuracy. Pixel accuracy metric is to simply figure out the percentage of correct predicted pixels in the image compared with ground truth. The pixel accuracy is commonly reported for each class separately or globally across all classes of prediction. Here we consider both of them with the names of **Pixel Accuracy** and **Class Accuracy**, which would be used in our experiments later. Pixel Accuracy simply measures the accuracy among every pixel in the image, whereas Class Accuracy does the evaluation for each semantic class and presents the mean value.

With this metric, we are evaluating a binary mask corresponding to its given ground truth. In details, a true positive (TP) case happens when a pixel is predicted correctly belonging to its given class. A true negative (TN) case is when a pixel is predicted correctly not belonging to the given class. Similarly, false positive (FP) and false negative (FN) cases respectively indicate incorrect prediction for belonging and not belonging to the considered class. Equation 4.3 shows how pixel accuracy is calculated.

$$\mathcal{P}ixelAccuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

However, this metric exists its disadvantage of providing misleading results when the class representation is too small in an image, as the measurement would be biased in mainly reporting the major case.

Mean Intersection-over-Union. To cope with the problem of pixel accuracy metric, we also used intersection-over-union (IoU). The IoU metric, also known as the Jaccard index, is used to quantify the percentage of the overlap between the target mask and the prediction output. The IoU measures the number of pixels in common of the two predicted mask and ground truth mask over the total number of pixels of the both. Equation 4.4 below presents how IoU is calculated.

$$\mathcal{I}oU = \frac{ground\_truth \cap prediction}{ground\_truth \cup prediction} \tag{4.4}$$

The IoU score is calculated for each class separately and then the average over all classes is combined to provide a mean IoU score of our semantic segmentation prediction. In this work, we consider **mIoU** as our target evaluation metric which we use to compare with other published results. Therefore, we mostly focus on the fluctuation of mIoU among experiments meanwhile the results of other metrics are provided to be used when in need.

Furthermore, to solve the case of class imbalance, we proposed to use **frequency weighted IoU (FWIoU)** to evaluate our model. This is an extension of mIoU. If there exists the domination of a semantic class in our dataset, that class needs to be weighed down to be equally important as other classes.

4.4.2 Experimental Results

Our experiments on semantic segmentation with self-training method were established with Panoptic Feature Pyramid Network model with ResNet101 as its backbone architecture (FPN-resnet101)[3]. Sections below present how efficient our changes take into account through series of experiments with various configurations.

4.4.2.1 Daytime Cityscapes Images Training

Experiment-1. Verifying self-training performance on daytime cityscapes dataset.

In this initial experiment, our FPN-resnet101 was trained on the data of standard Cityscapes images, which includes 2975 annotated daytime images in trainset and 500 annotated images in valset. Our extra unlabeled data was 701 images from all sets of Camvid dataset[4]. Our testset was 50 annotated images of Daytime Driving Test dataset (this is also the default testset for the rest of our experiments). The results of this initial experiments are shown in Table 4.3.

[3]Our experiments were setup on a single GPU GeForce GTX Titan X.

[4]http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/

[5]Self-training from scratch means that we used the initial model. Self-training from checkpoint means that we used the trained model of 1.1.

ID	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
1.1	FPN-res101-daytime-Cityscapes	73.6	**45.4**	27.5	59.0
1.2	FPN-res101-self-training-from-scratch[5]	76.3	43.9	27.1 (−0.4)	62.3
1.3	FPN-res101-self-training-from-ckpt	**76.8**	44.0	**29.0** (+1.5)	**62.7**

Table 4.3: Results of Segmentation Experiment-1. Verifying self-training performance on daytime cityscapes dataset.

The models in these experiments were trained completely on daytime images (including annotated training data and extra unlabeled data). Those results were tested on nighttime images and yielded the score of **29.0%** mIoU, along with **76.8%** pixel accuracy and **62.7%** FWIoU. Performing self-training on model **1.1**, we confirmed the statement of B Zoph et al [55] that self-training with extra unlabeled data always helps our model improved performance. Yet, note that self-training is helpful if we train from a checkpoint, which means that our model should have prior knowledge of fine annotated data, or it will ruins the results (as in model **1.2**) with a decrease of **0.4%** mIoU. Here, we observed an increase of **3.2%** (in accuracy), **1.5%** (in mIoU) and **3.7%** (in FWIoU), which are considerable.

The visualization showed that the model predicted with low performance of semantic segmentation. Particularly, pixels belonging to a semantic object is easily mislabeled and assigned to another class. Yet, totally we still catch good cases of segmenting the shape of human or traffic sign. Visualization of Experiment-1 is shown in Figure 4.11.

4.4.2.2 Daytime and Nighttime Images Training

Experiment-2. Narrowing down the distance between trainset and testset by adding generated nighttime cityscapes images together with self-training on true nighttime images.

In this experiment, we trained our FPN-res101 with both daytime and nighttime cityscapes images. The daytime images were the same as in Experiment-1, which was original Cityscapes trainset. While, the nighttime part was the result of translating daytime images to the domain of nighttime with the help of our GAN-based image translation module. Thus, images of the two parts shared their annotation. As a combination, our trainset in Experiment-2 consists of 5950 day-night images. We did the same with valset and resulted in total 1000 images in this set. With this configuration, we were in hope of teaching our model to predict segmentation better in nighttime mode. For self-training purpose, we leveraged 14937 nighttime cityscapes images (in NEXET dataset) as extra unlabeled data for this experiment. To our

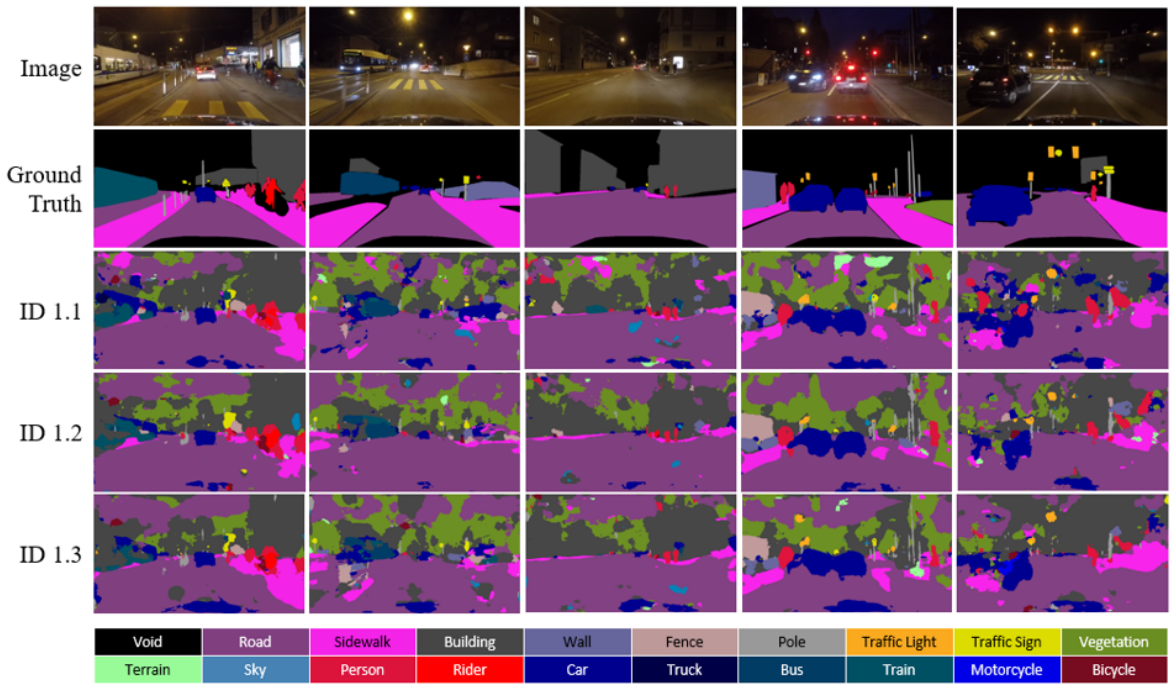

Figure 4.11: Visualization of Segmentation Experiment-1 results. ID 1.1 is the results of FPN-resnet101 trained on daytime images of Cityscapes; ID 1.2 is the results of self-training with the model from scratch; ID 1.3 is the results of self-training with checkpoint of ID 1.1. The unlabeled data in this experiment is 701 daytime cityscapes images of CamVid.

knowledge from Experiment-1, we do not perform self-training with the model from scratch from now on. Table 4.4 shows the results of this configuration.

ID	Testset	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
2.1	Origin	**FPN-res101-daynight**	**79.2**	**49.5**	**31.5**	**66.5**
2.2		FPN-res101-self-training-15k-from-ckpt-2.1	78.1	41.5	28.8 (−2.7)	63.8
2.3	Converted[6]	FPN-res101-daynight	74.1	42.5	25.2	60.7
2.4		FPN-res101-self-training-15k-from-ckpt-2.1	73.2	42.1	24.7 (−0.5)	58.8

Table 4.4: Results of Segmentation Experiment-2. Narrowing down the distance between trainset and testset by adding generated nighttime cityscapes images together with self-training on true nighttime images.

In this experiment, we tested on two version of testset, original Nighttime Driving Test and its daytime converted version. The converted version was generated by our images translation model, which was used to generate nighttime trainset and valset. To the aspect of original testset, we observed the results of training our model from scratch (**2.1**) better than self-training with 14937 extra unlabeled data. In details, the highest performance is **79.2%** pixel accuracy, **31.5%** mIoU and **66.5%** FWIoU, which improves **2.4%**, **2.5%** and **3.8%** of pixel accuracy, mIoU and FWIoU compared with the best of Experiment-1 (1.3), respectively. Specially, self-training in this case did not help but

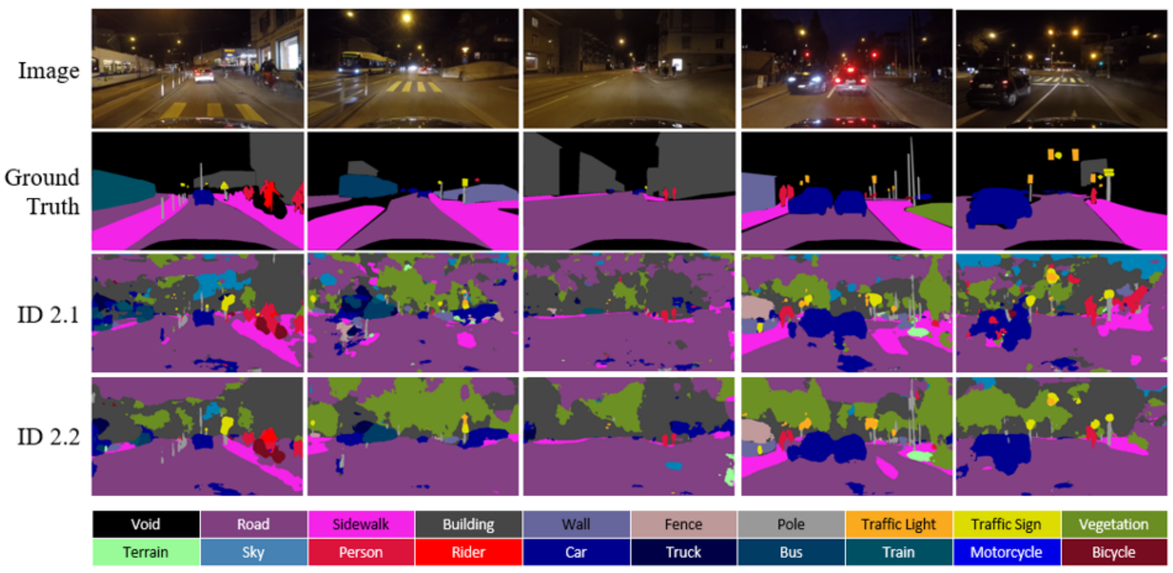

Figure 4.12: Visualization of Segmentation Experiment-2 results. ID 2.1 is the results of FPN-resnet101 trained on day and night images of Cityscapes; ID 2.2 is the results of self-training with checkpoint of ID 2.1. The unlabeled data in this experiment is around 15000 unlabeled nighttime cityscapes images of NEXET.

[6]Testset in Converted mode is the result of translating them to the daytime domain.

ruined the performance of our model. We explain this issue with the statement that the amount of extra unlabeled data dominated labeled data (with the ratio of around 2.5 times). In fact, pseudo-labels are assumed to be either correct or incorrect and with this too much pseudo images could confuse our model. **Therefore, we concluded that self-training could help improve our model, yet with suitable amount.** With the daytime converted version of testset, we also suffered from the same issue of unhelpful self-training, which strengthened our conclusion above. Furthermore, the results showed that the performance of our model on this converted testset is lower than on the original testset. This occurred as a result of our image translation module, which was not really good at translating between nighttime/daytime domains (this was mentioned in the section above). This means that there is a gap between our generated nighttime images and the real ones. In the next experiments, we would modify our image translation module to overcome this poor-quality translation. Visualization of Experiment-2 is illustrated in Figure 4.12.

4.4.2.3 Daytime and Nighttime Images Training with Perceptual Loss for Image Translation Module

Experiment-3. Improving segmentation performance by adding perceptual loss to maintain semantic features when translating images across domains.

To cope with the problem of the low performance of our image translation module, we proposed to apply perceptual loss to this stage. Then we trained our image translation model again with adding perceptual loss. In this experiment, our trainset also includes 2975 daytime images and 2975 nighttime images and our valset is also a combination of 500 daytime and 500 nighttime images. The difference from the previous experiment is that the quality of our nighttime domain improved significantly with the help of perceptual loss to our module. Here, we also tried to convert testset domain from nighttime domain to daytime domain for exemplary testing. To the aspect of extra unlabeled data, we also chose the same set as Experiment-2 with 14937 nighttime images from NEXET dataset. The details of this experiment are shown in Table 4.5.

Observed our experimental results, we figured out that perceptual loss played an important role in forming fine nighttime domain images. In details, the performance of our system measured by mIoU increased significantly an amount of **2.4%** (from 31.5% to 33.9%). This point demonstrated how serious the domain similarity was to train our segmentation model. To confirm our conclusion in Experiment-2 about self-training technique, we applied this technique to model in **3.1**. Once again, with the number of 14937 unlabeled images, we suffered from a downgrade result of **1.8%** mIoU

ID	Testset	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
3.1	Origin	FPN-res101-daynight	78.0	**51.3**	**33.9**	66.0
3.2		FPN-res101-self-training-15k-from-ckpt-3.1	**81.5**	43.0	32.1 (−1.8)	**69.0**
3.3	Converted	FPN-res101-daynight	75.2	46.1	29.3	62.7
3.4		FPN-res101-self-training-15k-from-ckpt-3.1	79.6	42.5	28.4 (−0.9)	66.8

Table 4.5: Results of Segmentation Experiment-3. Improving segmentation performance by adding perceptual loss to maintain semantic features when translating images across domains.

(from 33.9% to 32.1%), which strengthens our conclusion. However, self-training in this case helped to improve our model with the metrics of pixel accuracy and FWIoU. With the converted testset, we also received a low performance results compared with testing on the original testset. There, we concluded that **the converted testset to daytime domain did not help our system performance**. Yet, we will continue some experiment on this configuration. The visualization of these results is in Figure 4.13.

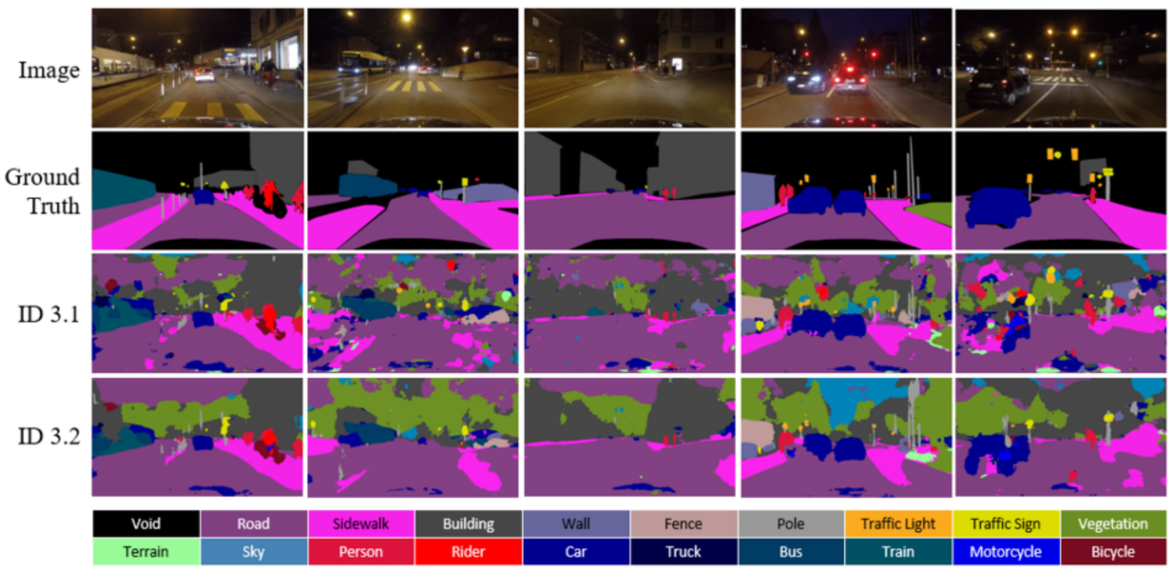

Figure 4.13: Visualization of Segmentation Experiment-3 results. ID 3.1 is the results of FPN-resnet101 trained on day and images of Cityscapes; ID 3.2 is the results of self-training with the model with checkpoint of ID 3.1. The image translation module has been added perceptual loss to maintain objects features.

Experiment-4. Improving self-training results by choosing extra unlabeled data with histogram-based method.

In the two previous experiments, we used 14937 nighttime cityscapes images from NEXET dataset as our extra unlabeled data. However, we found out that in this dataset, there exists many cases of nighttime images which are in extreme low light condition that is even difficult for human to understand (exemplary images in Figure 4.9). Furthermore, as we inferred from Experiment-1 and Experiment-3, when using 701 images of Camvid as our extra unlabeled data, our performance increased. In contrast, when using 14937 images from NEXET dataset as our extra unlabeled data, self-training ruined our model performance. Therefore, we tried to apply the ratio of our successful case in Experiment-1, which mean extra unlabeled data accounts for around 1/4 of our labeled training data. As a result, we set a threshold of histogram $15 < \alpha_{night} < 20$ with the weight of red channel is 0.5 (as mentioned in Section 4.2.2.3). Finally, we resulted in 1600 extra unlabeled images for self-training purpose. Besides, our trainset and valset maintain the same as in Experiment-3. In this experiment, we also tested converted daytime testset. Table 4.6 shows our results in details.

ID	Testset	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
3.1	Origin	FPN-res101-daynight	78.0	**51.3**	33.9	66.0
4.1		FPN-res101-self-training-1k6-HIS-from-ckpt-3.1	**80.4**	49.7	**34.2** (+0.3)	**68.2**
3.3	Converted	FPN-res101-daynight	75.2	46.1	29.3	62.7
4.2		FPN-res101-self-training-1k6-HIS-from-ckpt-3.1	77.6	46.1	29.8 (+0.5)	64.0

Table 4.6: Results of Segmentation Experiment-4. Improving self-training results by choosing extra unlabeled data with histogram-based method.

Thanks to the added perceptual loss, our nighttime translated images reached acceptable quality. And via our experimental results in the above report table, we perceived better performance of **34.2%** evaluated by mIoU metric, which increased slightly **0.3%** compared with model **3.1**. Besides, pixel accuracy and FWIoU increased **2.4%** and **2.2%**, respectively, meanwhile class accuracy reduced **1.6%**. Those results were evaluated on the original testset which are as our expectation. When testing on the converted testset, we again observed downgraded results on the four metrics. Thus, from here, we stopped experimenting on this converted testset. In Figure 4.14, we illustrates some exemplary images.

4.4.2.4 Only-night Images Training with Perceptual Loss for Image Translation Module

Experiment-5. Training segmentation model on target nighttime domain by using only nighttime cityscapes images.

In this configuration, our willing is to find out the performance of FPN-resnet101 trained on nighttime generated images only. As the previous experiments are trained with both day and night images, but the target domain is only night domain. However,

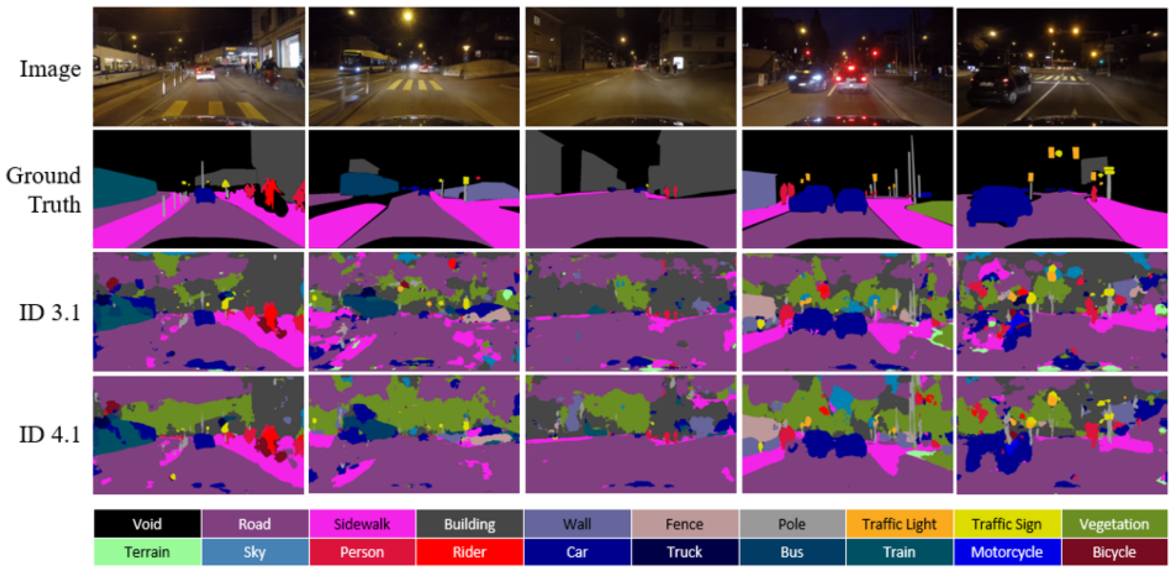

Figure 4.14: Visualization of Segmentation Experiment-4 results. ID 3.1 is the previous results trained on day and night images; ID 4.1 is the results of self-training with the model with checkpoint of ID 3.1. This experiment only tests the effect of unlabeled data. Here we pick 1600 images from 15000 unlabeled nighttime images of NEXET dataset by histogram-based method.

with this setup, we expect the model trained from scratch with this nighttime trainset would be lower than the previous results. There are **two reasons** for our expectation: our nighttime generated images still maintain difference compared with true night images and training on only nighttime would make the model lack of the ability of predicting light night images. Therefore, we set up our dataset with 2975 nighttime translated images only, compared with 5950 day-night images like previous experiments. The valset is also reduced to 500 nighttime images, compared with 1000 day-night images. Furthermore, we would use a checkpoint from the previous day-night trained model **3.1** to perform an extra refinement on nighttime images. The results are in Table 4.7.

ID	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
5.1	FPN-res101-onlynight	76.3	46.0	29.6	62.4
5.2	FPN-res101-morenight-from-ckpt-3.1	81.2	**48.8**	**34.7** (+0.8)	**68.7**
5.3	FPN-res101-self-training-1k6-HIS-from-ckpt-5.1	78.3	41.6	29.8 (+0.2)	64.2
5.4	FPN-res101-self-training-1k6-HIS-from-ckpt-5.2	**81.3**	46.0	33.3 (−1.4)	68.3

Table 4.7: Results of Segmentation Experiment-5. Training segmentation model on target nighttime domain by using only nighttime cityscapes images.

As our expectation, the model trained from scratch with nighttime images only achieved **29.6%** mIoU, particularly decreased **4.6%** compared with model **4.1**. And its self-training result in **5.3** increased slightly **0.2%** mIoU. Whereas, the model in checkpoint **3.1** with an extra training on nighttime images reached **34.7%** mIoU, increased **0.8%** compared to its checkpoint performance. This means that **an extra training on the target prediction domain helps improving the performance** of our system. However, we suffered from a decrease of **1.4%** mIoU when self-training with model **5.2**. The explanation for reduction in mIoU of this experiment is the checkpoint we used as the initial model was trained on day-night images, which brought the model prior knowledge in day-night images segmentation. Then, we performed an extra training on only nighttime images and a self-training on 1600 nighttime images, which might confuse and conflict with the prior knowledge of this model. To be honest, this experiments brought the finest score from model **5.2** with **34.7%** mIoU. Visualization of this run is illustrated in Figure 4.15.

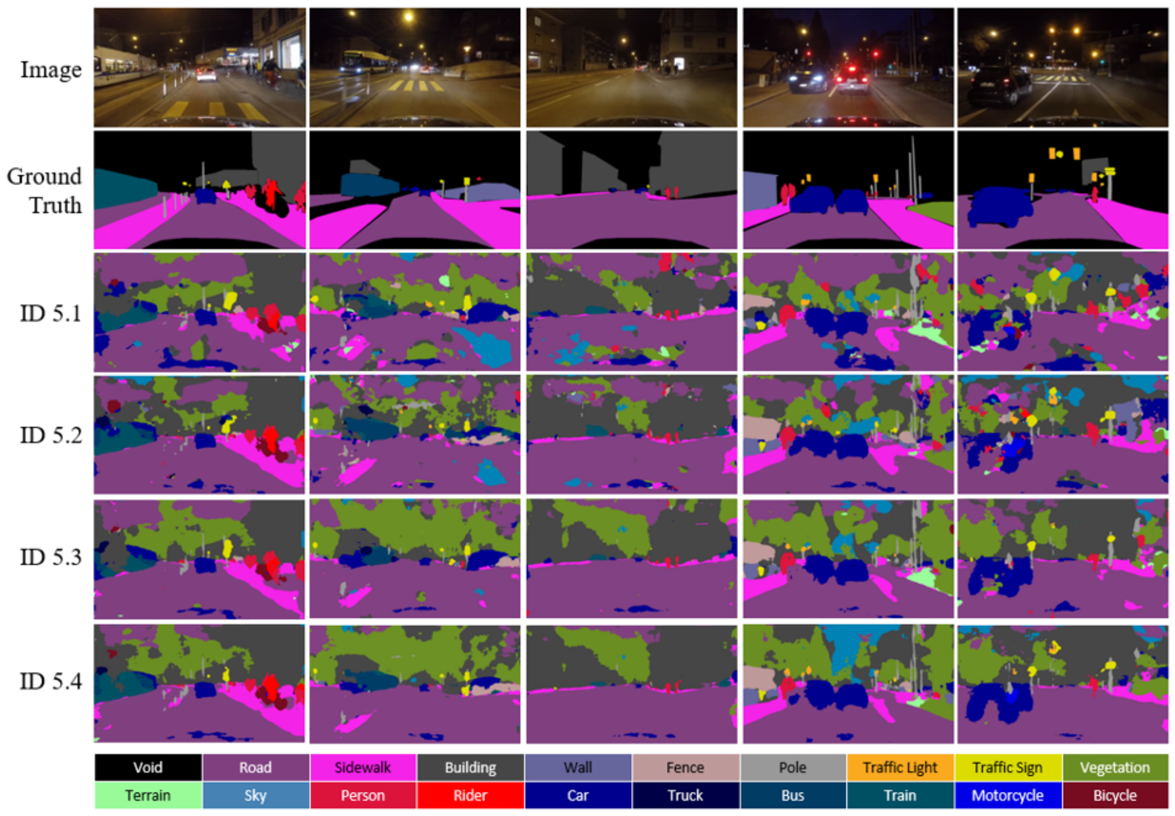

Figure 4.15: Visualization of Segmentation Experiment-5 results. ID 5.1 is the results of FPN-resnet101 trained on only nighttime cityscapes images (generated by GAN); ID 5.2 is the model in ID 3.1 trained with more nighttime images; ID 5.3 and 5.4 are the results of self-training on 1600 unlabeled nighttime images selected by histogram-based method with the checkpoints of ID 5.1 and 5.2, consecutively.

4.4.2.5 Daytime and Nighttime Images Training with Focal Loss

Experiment-6. Trying focal loss to train segmentation model.

In this experiment, we replaced the cross entropy loss function of our semantic segmentation model with focal loss (as mentioned in Section 3.3.2). Our target is to test whether the focal loss can help better converge our segmentation model by overcoming problems of cross entropy loss. Our dataset in this run is similar to those of Experiment-4, which is day-night trainset and valset. Our experimental results are presented in Table 4.8.

ID	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
3.1	FPN-res101-daynight-CE	78.0	**51.3**	33.9	66.0
4.1	FPN-res101-self-training-1k6-HIS-from-ckpt-3.1-CE	**80.4**	49.7	**34.2** (+0.3)	**68.2**
6.1	FPN-res101-daynight-FL	74.4	40.9	26.9	60.0
6.2	FPN-res101-self-training-1k6-HIS-from-ckpt-6.1-FL	76.0	43.3	28.3 (+1.4)	61.8

Table 4.8: Results of Segmentation Experiment-6. Trying focal loss to train segmentation model.

We observed that focal loss took more time to converge our segmentation model. To be specific, we set up the same number of epochs for each of our experiments and the model trained with focal loss showed lower results than the same model trained with cross entropy loss (model **3.1**). However, once again we declared the help of self-training when increasing **1.4%** mIoU of our system performance as well as slight improvements on other evaluation metrics. With this experiment, we concluded that **focal loss did not have great effect on our semantic segmentation model compared to cross entropy loss.** The visualization is shown in Figure 4.16.

4.4.2.6 Daytime and Nighttime Images Training with FID-based Method for Extra Unlabeled Data

Experiment-7. Improving self-training performance by using FID-based method to select extra unlabeled data and testing our proposed loss function.

In Experiment-4, we picked out 1600 true nighttime images from NEXET dataset with histogram-based method and achieved good results (model **4.1**). Yet, we realized that choosing nighttime images based on image histogram did not leverage the semantic information of images, which could result in unsuitable nighttime images. To be specific, as histogram does not know the semantic aspects, it might consider an image with many black cars or black buildings as a nighttime domain. Thus, we proposed to use FID-based method (as mentioned in 4.2.2.3) to choose around 1600 images again

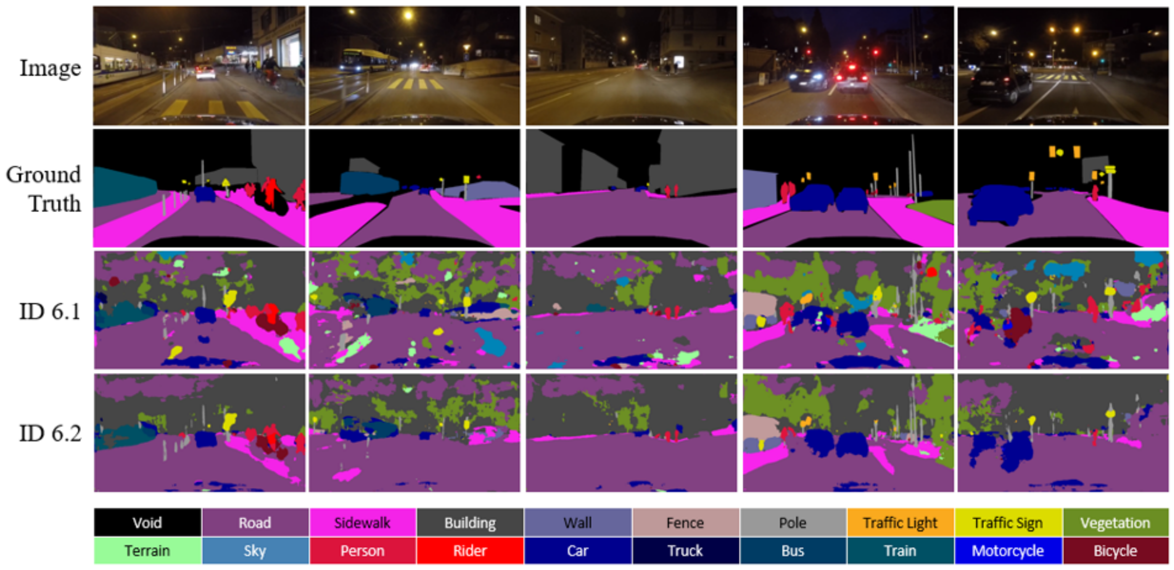

Figure 4.16: Visualization of Segmentation Experiment-6 results. ID 6.1 is the results of FPN-resnet101 trained on day and night cityscapes images with focal loss; ID 6.2 is the results of self-training on 1600 unlabeled nighttime images and also trained with focal loss. This experiment was held to compare the performance of focal loss and cross entropy loss among models.

(from NEXET nighttime images). Our trainset and valset were day-night images as we did set up in previous experiments. Instead of training our FPN-resnet101 from scratch, we chose to take model **3.1** (which prior knowledge on day-night images) as our checkpoint for this experiment. Our expectation is the increase in the result of self-training technique. Moreover, we also tried our proposed combined loss function in this experiment to observe its effectiveness. The reported results are in Table 4.9.

ID	**Configuration**	**Accuracy**	**Class Accuracy**	**mIoU**	**FWIoU**
3.1	FPN-res101-daynight-CE	78.0	**51.3**	33.9	66.0
7.1	FPN-res101-self-training-1k6-FID-from-ckpt-3.1-CE	**84.3**	49.5	38.8 (+4.9)	**73.4**
7.2	FPN-res101-self-training-1k6-FID-from-ckpt-3.1-CL	83.8	50.5	**39.3** (+5.4)	72.4
7.3	FPN-res101-self-training-1k6-HIS-from-ckpt-3.1-CL	81.5	45.2	33.1 (−0.8)	68.5

Table 4.9: Results of Segmentation Experiment-7. Improving self-training performance by using FID-based method to select extra unlabeled data and testing our proposed loss function.

The reported results proved that **choosing extra unlabeled data with FID-based method helps to significantly improve our model performance**. With the same cross entropy loss function in model **7.1**, we observed a big jump of **4.9%** in mIoU compared to its checkpoint in model **3.1**. With FID-based method, we leveraged deep features among nighttime images to pick out those with the same distribution as much as possible. This Experiment-7 leads to a conclusion that **the performance of segmentation model depends on the data domain distribution**. From another viewpoint, using FID-based method to choose extra unlabeled data is a technique to narrow down the distance between the two domains of trainset and testset.

To the aspect of using our proposed combined loss function, we achieved our expectation. Among the two model of **7.1** (using cross entropy loss - CE) and **7.2** (using combined loss - CL), the model with combined loss yields higher mIoU value of **39.3%**, which improved **5.4%** compared to its checkpoint and improved **0.5%** compared to the model trained with cross entropy loss. This confirms that our **combined loss function makes use of its components specifications and assists to better train the model**. On the other hand, model **7.3** modified the declaration of how effective FID-based method is when training with combined loss function. Figure 4.17 shows exemplary results of this experiment.

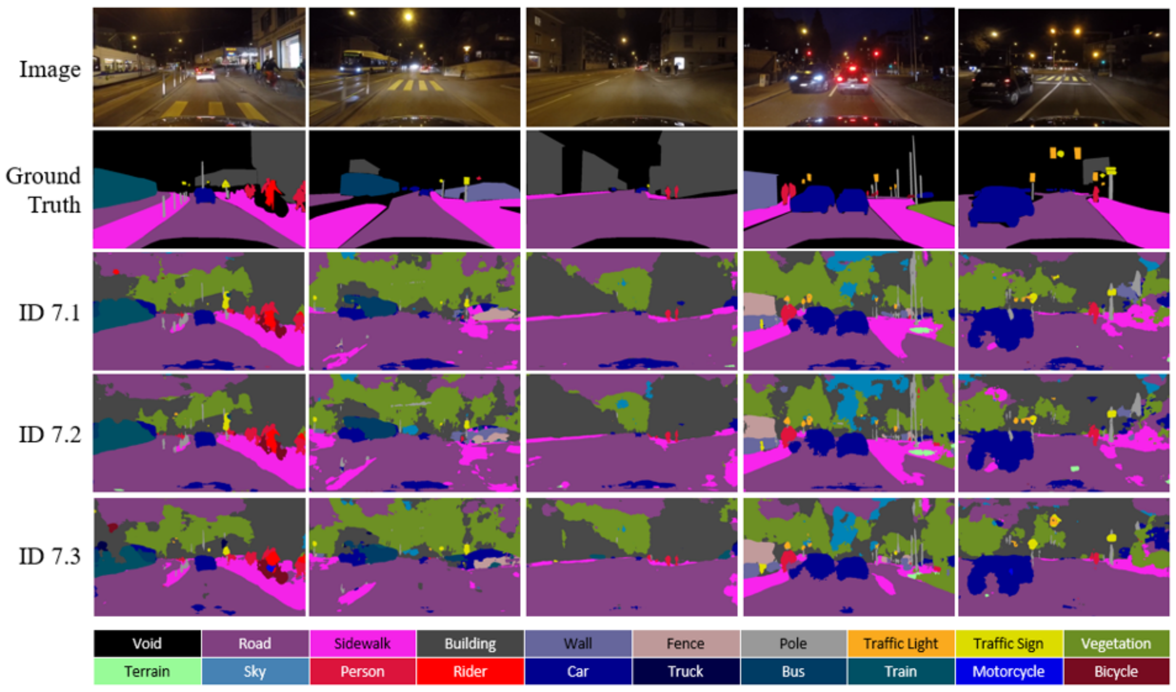

Figure 4.17: Visualization of Segmentation Experiment-7 results. ID 7.1 is the results of self-training on around 1600 unlabeled nighttime images chosen by FID-based method and trained with cross entropy loss on checkpoint ID 3.1; ID 7.2 is similar to ID 7.1 but with our proposed combined loss; ID 7.3 is to compared histogram-based with FID-based methods. FID-based method together with our proposed loss function yields the finest score.

Experiment-8. Trying the combo of FID-based method, perceptual loss and our proposed loss function.

This experiment is performed to verify if more annotated nighttime image training along with self-training and FID-based method help our segmentation model. In details, we base on the day-night images training of model **3.1** and refine that model with annotated nighttime images which results in model **5.2**. From the checkpoint of model **5.2**, we did self-training with extra unlabeled data chosen by histogram based method and we denote this model as **8.1**. In this experiment, we combined all modifications together to check the total efficiency. Our modifications include combined loss function, FID-based method to choose extra unlabeled data and more annotated nighttime image training. The dataset components are kept stable as in Experiment-7. The results are shown in Figure 4.10.

ID	Configuration	Accuracy	Class Accuracy	mIoU	FWIoU
3.1	FPN-res101-daynight-CE	78.0	**51.3**	33.9	66.0
5.2	FPN-res101-morenight-from-ckpt-3.1	81.2	48.8	34.7 (+0.8)	68.7
8.1	FPN-res101-self-training-1k6-HIS-from-ckpt-5.2-CE	83.2	48.9	37.8 (+3.9)	71.2
8.2	FPN-res101-self-training-1k6-FID-from-ckpt-8.1-CE	83.4	49.1	39.5 (+5.6)	**71.4**
8.3	FPN-res101-self-training-1k6-FID-from-ckpt-8.1-CL	**83.7**	51.1	**40.7** (+6.8)	**71.9**

Table 4.10: Results of Segmentation Experiment-8. Trying the combo of FID-based method, perceptual loss and our proposed loss function.

Our base model in this experiment is model **8.1**, from that checkpoint, we perform the combination of modifications. Model **8.2** and **8.3** is to test the effectiveness of cross entropy loss and combined loss functions, respectively. The reported results show that the combination of all modifications (with combined loss) achieves the finest result of **40.7%** mIoU, increases **6.8%** compared to the checkpoint of day-night image training in model **3.1**. In comparison with the very first initial experiment in model **1.1**, our improvement in mIoU increase a number of **13.2%**. This experiment proved that **our combined loss function actually help to better train our segmentation rather than using only each of its components**. To end up, we did eight sets of experiments to significantly improve the baseline of FPN-resnet101 from **27.5%** mIoU to **40.7%**. The visualization is in Figure 4.18 in order to compare segment maps among results in Experiment-8 and the initial results.

4.4.3 Lessons from Series of Experiments

After performing such series of experiments above, we come up with lessons for each experiment:

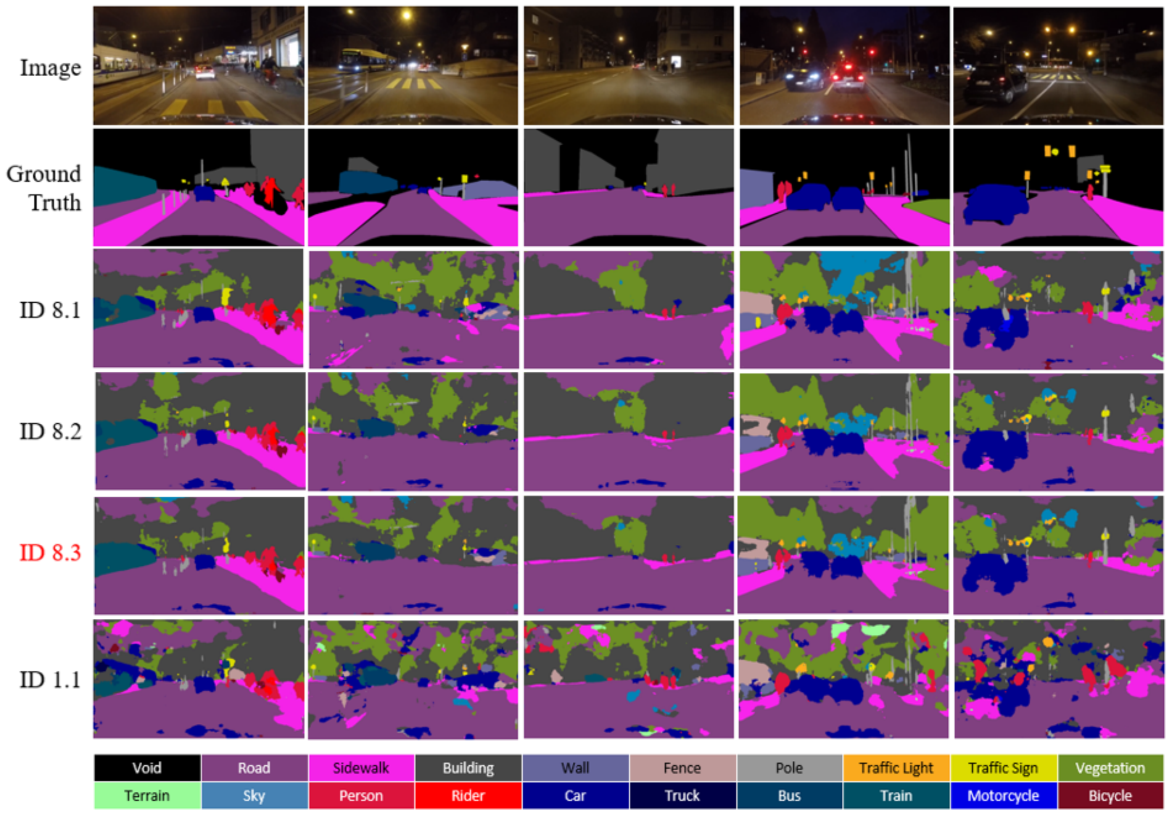

Figure 4.18: Visualization of Segmentation Experiment-8 results. ID 8.1 is the results of self-training on around 1600 unlabeled nighttime images chosen by histogram-based method from checkpoint ID 5.2; ID 8.2 is similar but unlabeled data is chosen with FID-based method, both 8.1 and 8.2 use cross entropy loss; ID 8.3 is the results of self-training on around 1600 unlabeled nighttime images from FID-based method with the checkpoint of ID 8.1 and the help of our proposed combined loss function. ID 8.3 is our finest performance model.

- Experiment-1: Verifying self-training performance on daytime cityscapes dataset. The difference of domain distribution between trainset and testset leads to poor performance of the model. Performance of a model when self-training from a checkpoint is actually better than from scratch.

- Experiment-2: Narrowing down the distance between trainset and testset by adding generated nighttime cityscapes images together with self-training on true nighttime images. Minimizing the difference between trainset and testset distribution helps improve the performance of the model. Self-training does not always help when the amount of unlabeled data dominates labeled data.

- Experiment-3: Improving segmentation performance by adding perceptual loss to maintain semantic features when translating images across domains. Perceptual loss helps better translate image domain from daytime to nighttime when learning to represent high level features, which benefits the segmentation training process.

- Experiment-4: Improving self-training results by choosing extra unlabeled data with histogram-based method. Self-training contributes to the success of the model, but with suitable amount of unlabeled data.

- Experiment-5: Training segmentation model on target nighttime domain by using only nighttime cityscapes images. An extra training on the target prediction domain helps improving the performance.

- Experiment-6: Trying focal loss to train segmentation model. Focal loss did not have great effect on our semantic segmentation model compared to cross entropy loss.

- Experiment-7: Improving self-training performance by using FID-based method to select extra unlabeled data and testing our proposed loss function. Choosing extra unlabeled data with FID-based method helps to significantly improve our model performance. Our proposed loss function benefits the training process.

- Experiment-8: Trying the combo of FID-based method, perceptual loss and our proposed loss function. The combination of training on day and night images together with extra training on nighttime and FID-based method and our proposed loss function helps better train the segmentation model.

Chapter 5

Conclusion

5.1 Thesis Achievements

More and more applications of semantic segmentation are systematically deployed with an aim at leveling-up the life standard. As mentioned in Chapter 1, semantic image segmentation plays a crucial role in helping autonomous vehicles to highly adapt diverse conditions: illuminations, perspectives and so on. In this thesis, we totally focus on semantic image segmentation with nighttime cityscapes images as the input and our framework outputs the segmentation maps corresponding to the semantic objects.

To perform segmentation on nighttime images, this task faces numerous difficulties, especially data. The biggest problem is lack of nighttime training annotated images. Hence, it comes up with the inspiration of leveraging the available annotated daytime data to help the model adapt with nighttime using domain adaptation.

To be more detailed, we propose a novel framework to adapt the segmentation model from daytime domain to nighttime domain by applying recent well-known method of GAN and self-training technique. In chapter 3, we describe our two components including Day2Night Image Translation Component and Semantic Image Segmentation Component. Another striking feature is that we applied novel self-training technique to improve the performance of the whole framework remarkably. After observing the failure results, we propose a new loss function, which enhances the results significantly. In chapter 4, we illustrate the whole experimental results and conclude the knowledge from each improvement step. We also modify several kinds of dataset by histogram and FID to evaluate its influence of data distribution to the self-training technique.

In the processing of this dissertation, we have gained a lot of insights. In a nutshell, our three main contributions are listed below:

- Propose a complete framework with domain adaptation method for semantic image segmentation in the dark with making use of GAN-based methods and self-training technique.
- Propose a loss function that optimizes the semantic image segmentation model performance.
- Build a nighttime cityscapes dataset by GAN-based method for the task of semantic segmentation.

5.2 Future Work

This section presents potential future work of our thesis. We divide the future improvement into two components corresponding to the two phases in our proposed framework.

5.2.1 Improving GAN-based Method

Image Enhancement. There are multiple ways to improve the results of GAN-based component, image enhancement is a simple one. Via observing some failure cases, we realise that there are still some limitations. Firstly, some translated images seem to be extremely dark. Secondly, some results are likely to have bad illumination when transferring from daytime to nighttime. To address the two mentioned problems, we can apply image enhancement method to adjust the results to the target domain.

Expansion of Objects in Dataset. In this thesis, we totally focus on the task of road scene segmentation in low-light condition. There is still a wide range of fields we can participate in. Hence, the expansion of kind of objects in the dataset is one way to enlarge the objectives of the whole framework.

5.2.2 Improving Semantic Image Segmentation Component

Model Performance. In this work, we base on panoptic feature pyramid networks to do the task of semantic image segmentation. Despite the fact that this model can extract multi-scale features and leverage both low and high semantic features of the input images, there exists many limitations. The remaining task is to refine the network architecture or simply to try our framework with other segmentation model that satisfies complicated requirements.

Nighttime Video Segmentation. This thesis aims at solving the task of nighttime cityscapes images segmentation on a target image. However, we understand the demand of processing this framework on video or even real-time video. Autonomous vehicles require immediate responds to the surroundings during its operation to guarantee the safe. Thus, one more to do task is to refine our framework to adapt real-time video object segmentation.

Pseudo Labels Inference in Self-training. In self-training stage, we use a trained segmentation model to infer pseudo labels from extra unlabeled data. At the present, our framework leverages all the pseudo labels without caring whether they are really good. This step can be improved by generating confidence maps along with pseudo labels. Then, there is a threshold of confidence to keep or ignore the pseudo labels. This approach is believed to improve the quality of pseudo labels generated for extra data. (Thesis Defense Update)

Appendix A

Publication

Journal Paper:

[1] Xuan-Duong Nguyen, Anh-Khoa Nguyen Vu, **Thanh-Danh Nguyen**, **Nguyen Phan**, Bao-Duy Duyen Dinh, Nhat-Duy Nguyen, Tam V. Nguyen, Vinh-Tiep Nguyen, Duy-Dinh Le: *Adaptive Detection-Tracking-Counting Framework for Multi-Vehicle Motion Counting*, Image and Vision Computing – IMAVIS (ISI Q1), 2021. (under review)

Bibliography

[1] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[2] 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.

[3] 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.

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

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

[6] Se Woon Cho, Na Rae Baek, Ja Hyung Koo, Muhammad Arsalan, and Kang Ryoung Park. Semantic segmentation with low light images by modified cyclegan-based image enhancement. *IEEE Access*, 8:93561–93585, 2020.

[7] 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.

[8] Dengxin Dai and Luc Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3819–3824. IEEE, 2018.

[9] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raska. Deepglobe 2018: A challenge to parse the earth through satellite images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 172–17209. IEEE, 2018.

[10] Donald Geman and Bruno Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[12] Rezwanul Haque, Milon Islam, Kazi Saeed Alam, Hasib Iqbal, and Ebrahim Shaik. A computer vision based lane detection approach. *International Journal of Image, Graphics & Signal Processing*, 11(3), 2019.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[14] 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.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[17] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[18] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[20] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Jun Li, Li Feng, Xin Zhang, Xiaodong Hu, et al. Adaptive scale selection for multiscale segmentation of satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(8):3641–3651, 2017.

[24] Xiaomeng Li, Lequan Yu, Hao Chen, Chi-Wing Fu, Lei Xing, and Pheng-Ann Heng. Transformation-consistent self-ensembling model for semisupervised medical image segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[25] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[26] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[27] 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.

[28] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 82–92, 2019.

[29] Liangliang Liu, Jianhong Cheng, Quan Quan, Fang-Xiang Wu, Yu-Ping Wang, and Jianxin Wang. A survey on u-shaped networks in medical image segmentations. *Neurocomputing*, 409:244–258, 2020.

[30] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

[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] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[33] Rohit Mohan and Abhinav Valada. Efficientps: Efficient panoptic segmentation. *arXiv preprint arXiv:2004.02307*, 2020.

[34] Dudi Nassi, Raz Ben-Netanel, Yuval Elovici, and Ben Nassi. Mobilbye: Attacking adas with camera spoofing. *arXiv preprint arXiv:1906.09765*, 2019.

[35] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[36] Horia Porav, Tom Bruls, and Paul Newman. Don't worry about the weather: Unsupervised condition-dependent domain adaptation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 33–40. IEEE, 2019.

[37] Salah Rifai, Y. Bengio, Aaron Courville, Pascal Vincent, and Mehdi Mirza. Disentangling factors of variation for facial expression recognition. pages 808–822, 10 2012.

[38] E. Romera, L. M. Bergasa, K. Yang, J. M. Alvarez, and R. Barea. Bridging the day and night domain gap for semantic segmentation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1312–1318, 2019.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[40] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *European Conference on Computer Vision*, pages 86–103. Springer, 2018.

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[42] MS Sonawane and CA Dhawale. A brief survey on image segmentation methods. *International Journal of Computer Applications*, 975:8887, 2015.

[43] Lei Sun, Kaiwei Wang, Kailun Yang, and Kaite Xiang. See clearer at night: towards robust nighttime semantic segmentation through day-night image conversion. In *Artificial Intelligence and Machine Learning in Defense Applications*, volume 11169, page 111690A. International Society for Optics and Photonics, 2019.

[44] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020.

[45] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[46] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[47] Yutong Xie, Jianpeng Zhang, Hao Lu, Chunhua Shen, and Yong Xia. Sesv: Accurate medical image segmentation by predicting and correcting errors. *IEEE Transactions on Medical Imaging*, 40(1):286–296, 2020.

[48] Yuanyou Xu, Kaiwei Wang, Kailun Yang, Dongming Sun, and Jia Fu. Semantic segmentation of panoramic images using a synthetic dataset. In *Artificial Intelligence and Machine Learning in Defense Applications*, volume 11169, page 111690B. International Society for Optics and Photonics, 2019.

[49] Kailun Yang, Luis M Bergasa, Eduardo Romera, and Kaiwei Wang. Robustifying semantic cognition of traversability across wearable rgb-depth cameras. *Applied optics*, 58(12):3141–3155, 2019.

[50] Ruoyu Yang, Zia U Ahmed, Urs C Schulthess, Mustafa Kamal, and Rahul Rai. Detecting functional field units from satellite images in smallholder farming systems using a deep learning based computer vision approach: A case study from bangladesh. *Remote Sensing Applications: Society and Environment*, 20:100413, 2020.

[51] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[52] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018.

[53] 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.

[54] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[55] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in Neural Information Processing Systems*, 33, 2020.