Semantic Segmentation Considering Image Degradation, Global Context, and Data Balancing

Dazhou Guo

Follow this and additional works at: https://scholarcommons.sc.edu/etd

Part of the Computer Sciences Commons

Recommended Citation

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact dillarda@mailbox.sc.edu.
SEMANTIC SEGMENTATION CONSIDERING IMAGE DEGRADATION, GLOBAL CONTEXT, AND DATA BALANCING

by

Dazhou Guo

Bachelor of Electronic Engineering
Dalian University of Technology China 2008

Master of Information and Informatic Engineering
Tianjin University China 2010

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Computer Science and Engineering
College of Engineering and Computing
University of South Carolina
2019

Accepted by:

Song Wang, Major Professor
Michael N. Huhns, Committee Member
Lannan Luo, Committee Member
Yan Tong, Committee Member
Xiaofeng Wang, Committee Member
Cheryl L. Addy, Vice Provost and Dean of the Graduate School
ACKNOWLEDGMENTS

I would like to acknowledge the support and encouragement I received during my doctoral research. I am incredibly grateful for those of you who shared with me personal and professional supports. I thank you for helping me during the journey.

First, I want to thank my adviser Dr. Song Wang. He has taught me how to become a good researcher at work. I appreciate his contributions of time and ideas to make Ph.D experience stimulating. To my dissertation committee: Dr. Michael Huhns, Dr. Lanan Luo, Dr. Yan Tong, and Dr. Xiaofeng Wang, I appreciate your time that you have spent on reading my dissertation and invaluable advice for my research. It is an honor to have you as my committee members.

Second, I want to thank my colleagues and friends in the Computer Vision Lab and the Aphasia Lab including (in alphabetical order): Dr. Alexandra Basilakos, Dr. Yu Cao, Dr. Sara Grace Dalton, Dr. Xiaochuan Fan, Dr. Paul Fillmore, Dr. Julius Fridriksson, Hao Guo, Dr. Shizhong Han, Dr. Isabel Hubbard, Dr. Yuewei Lin, Dr. Ping Liu, Dr. Zibo Meng, Yang Mi, Yanting Pei, Dr. Jessica Richardson, Dr. Chris Rorden, Dr. Kimberly Smith, Shaoyue Song, Dr. Helga Thors, Haozhou Yu, Dr. Hongkai Yu, Dr. Youjie Zhou, Kang Zheng. Thank all of you for providing support, inspiration, mentoring, and motivation along the way.

Finally, my most sincere gratitude goes to my family. I give my heartfelt thanks to my parents and Shuyun. I cannot be here without their unlimited supports and encouragements. They are the essential parts of my life and are the key fundamental to help me and keep me going.
Abstract

Recently, semantic segmentation – assigning a categorical label to each pixel in an image – plays an important role in image understanding applications, e.g., autonomous driving, human-machine interaction and medical imaging. Semantic segmentation has made progress by using the deep convolutional neural networks, which are surpassing the traditional methods by a large margin. Despite the success of the deep convolutional neural networks (CNNs), there remain three major challenges.

The first challenge is how to segment the degraded images semantically, i.e., degraded image semantic segmentation. In general, image degradations increase the difficulty of semantic segmentation, usually leading to decreased segmentation accuracy. While the use of supervised deep learning has substantially improved the state-of-the-art of semantic segmentation, the gap between the feature distribution learned using the clean images and the feature distribution learned using the degraded images poses a major obstacle to degraded image semantic segmentation. We propose a novel Dense-Gram Network to more effectively reduce the gap than the conventional strategies in segmenting degraded images. Extensive experiments demonstrate that the proposed Dense-Gram Network yields state-of-the-art semantic segmentation performance on degraded images synthesized using PASCAL VOC 2012, SUNRGBD, CamVid, and CityScapes datasets.

The second challenge is how to embed the global context into the segmentation network. As the existing semantic segmentation networks usually exploit the local context information for inferring the label of a single pixel or patch, without the global context, the CNNs could miss-classify the objects with similar color and shapes. In
this thesis, we propose to embed the global context into the segmentation network using object’s spatial relationship. In particular, we introduce a boundary-based metric that measures the level of spatial adjacency between each pair of object classes and find that this metric is robust against object size induced biases. By enforcing this metric into the segmentation loss, we propose a new network, which starts with a segmentation network, followed by a new encoder to compute the proposed boundary-based metric, and then train this network in an end-to-end fashion for semantic image segmentation. We evaluate the proposed method using CamVid and CityScapes datasets and achieve favorable overall performance and a substantial improvement in segmenting small objects.

The third challenge of the existing semantic segmentation network is the performance decrease induced by data imbalance. At the image level, one semantic class may occur in more images than another. At the pixel level, one semantic class may show larger size than another. Classic strategies such as class re-sampling or cost-sensitive training could not address these data imbalances for multi-label segmentation. Here, we propose a selective-weighting strategy to consider the image- and pixel-level data balancing simultaneously when a batch of images are fed into the network. The experimental results on the CityScapes and BRATS2015 benchmark datasets show that the proposed method can effectively improve the performance.
# Table of Contents

### Acknowledgments

vii

### Abstract

iv

### List of Tables

ix

### List of Figures

xiii

### Chapter 1 Introduction

1

1.1 Challenges

2

1.2 Scope of the Research

7

1.3 Proposed Approaches

9

1.4 Structure of the Thesis

11

### Chapter 2 Background

12

2.1 Convolutional Neural Networks Basics

12

2.2 Convolutional Neural Networks Backbones

19

2.3 Semantic Segmentation Networks

23

2.4 Datasets

28

### Chapter 3 Literature Review

32

3.1 An Overview of Feature-based Learning Approaches

32
# Table of Contents

3.2 Related Works of Degraded Image Segmentation ........................................ 33
3.3 Related Works of Global Context Embedding ........................................ 35
3.4 Related Works of Data Balancing ......................................................... 37

Chapter 4 Degraded Image Segmentation with Dense-Gram Network ............... 39
4.1 Motivation .................................................................................. 39
4.2 Method .................................................................................... 43
4.3 Experiments ............................................................................. 50
4.4 Conclusion .............................................................................. 63

Chapter 5 Global Context Embedding with Inter-class Shared Boundary Based Encoder ......................................................... 64
5.1 Motivation ................................................................................ 64
5.2 Method ................................................................................... 67
5.3 Experiments ............................................................................ 74
5.4 Conclusion .............................................................................. 82

Chapter 6 Data Balancing with Selective Weighting .................................... 83
6.1 Motivation ................................................................................ 83
6.2 Method .................................................................................... 85
6.3 Experiments ............................................................................ 90
6.4 Conclusion .............................................................................. 98

Chapter 7 Conclusion ........................................................................ 100
7.1 Summary ................................................................................ 100
List of Tables

Table 2.1  ImageNet top-1 and top-5 error rates. The networks are trained and validated using PyTorch. ......................................................... 20

Table 4.1  The mIoUs (in percentage) of segmenting degraded images using: 1) Baseline networks fine-tuned using the degraded images (*+fine-tune); 2) Baseline networks trained using both clean and degraded images (*+C&D); 3) DGN trained using the degraded images (*+DGN); 4) DGN trained using paired clean and degraded images (*+DGN+C&D). “Clean” denotes the mIoUs on the clean images. The five degradation degrees are denoted using \( d_1, d_2, d_3, d_4, \) and \( d_5 \), respectively. The numbers with the better and best performance are highlighted in blue and red, respectively. .................... 49

Table 4.2  The mIoUs (in percentage) of segmenting clean images using 1) fine-tuned baseline; 2) trained DGN; 3) trained DGN+C&D. The five degradation degrees are denoted using \( d_1, d_2, d_3, d_4, \) and \( d_5 \), respectively. The numbers with the best performance are highlighted in red. .......................................................... 50

Table 4.3  The mIoUs (in percentage) of segmenting degraded images using: 1) Pre-trained model tested using the restored images; 2) Baseline network fine-tuned using the restored images (*+fine-tune); 3) DGN trained using the degraded images (DGN); 4) DGN trained using the restored images (*+DGN). The five degradation degrees are denoted using \( d_1, d_2, d_3, d_4, \) and \( d_5 \), respectively. The numbers with better and the best performance are highlighted in blue and red, respectively. ....................... 52

Table 4.4  The mIoUs (in percentage) of segmenting degraded images using: 1) DGN; 2) Direct feature maps matching (DGN-MSE); 3) Linear kernel (DGN-Linear); 4) Gaussian kernels (DGN-Gaussian); 5) Layer-wise Gram matrices matching (DGN-Layerwise). For each degradation effect, the degradation degree is increased from left to right. The numbers with the best performance are highlighted in red. .................................................. 55
Table 4.5 The mIoUs (in percentage) of segmenting degraded images using different $\lambda$ in Eq. (4.5), default $\lambda = 1 \times 10^{-1}$. The five degradation degrees are denoted using $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively. The numbers with the best performance are highlighted in red. ................................................................. 56

Table 4.6 The mIoUs (in percentage) of segmenting degraded images when the dense-Gram matching begins at the 2nd, 3rd, 4th, 5th, and 6th convolutional blocks. For each degradation effect, the degradation degree is increased from left to right. The numbers with the best performance are highlighted in red. ................................................................. 57

Table 4.7 The mIoUs (in percentage) of segmenting degraded images using different $\rho$. Let “EWC” denote the segmentation network that employs the EWC module. The five degradation degrees are denoted using $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively. The numbers with better and the best performance are highlighted in blue and red, respectively. ................................................................. 57

Table 4.8 The mIoUs (in percentage) of segmenting real haze images. The five degradation degrees are denoted using $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively. The numbers with the best performance are highlighted in red. ................................................................. 62

Table 5.1 Detailed configuration of the proposed ISBEncoder architecture. We use the image size of $360 \times 480$ for demonstration. ............... 71

Table 5.2 The comparison results of small object classes (left) and large object classes (right) on CamVid testing dataset (in percentage). For each object class, the numbers with better and the best performance are highlighted in blue and red, respectively. ............... 73

Table 5.2 The comparison results of small object classes (top) and large object classes (bottom) on CityScapes testing dataset (in percentage). For each object class, the numbers with better and the best performance are highlighted in blue and red, respectively. 75

Table 5.3 The ISBEncoder mean square errors of the small object classes (MSE$_S$), the large object classes (MSE$_L$), and all classes (MSE) on CamVid and CityScapes testing datasets. .............................. 78
Table 5.4 The comparison results of small object classes (left) and large object classes (right) using the proposed pipeline with different hyperparameter $\lambda$ (in Eq. (5.4)) on CamVid testing dataset (in percentage). The baseline segmentation network is FCN-8s. For each object class, the numbers with the best performance are highlighted in red.

Table 5.5 The comparison results of small object classes (top) and large object classes (bottom) using proposed pipeline with different hyper-parameter $\lambda$ (in Eq. (5.4)) on CityScapes testing dataset (in percentage). The baseline segmentation network is PSPNet. For each object class, the numbers with the best performance are highlighted in red.

Table 6.1 Average performance (in percentage) of the CityScapes testing dataset. Let $\dagger$ denote the proposed method that balances the data for the whole dataset instead of for each batch. Compared with the baseline networks, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

Table 6.2 Average performance (in percentage) of the BRATS 2015 testing dataset. Compared with the baseline networks, for each type of tumor, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

Table 6.3 Average performance (in percentage) of the CityScapes testing dataset. The baseline network is FCN-8s with WCE-based data balancing method. The numbers with the best performance are highlighted in red.

Table 6.4 Average performance (in percentage) of the CityScapes testing dataset. The networks are trained with or without the image-level balancing (IB) method or the pixel-level balancing (PB) method. Compared with the baseline networks, for each object class, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

Table 6.5 Average performance (in percentage) of the BRATS 2015 testing dataset. The networks are trained with or without the IB method or the PB method. Compared with the baseline networks, for each object class, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.
Table 6.6  Average performance (in percentage) of the BRATS 2015 testing dataset. The baseline network is DeepMedic, where the default batch size is 50. The networks are trained using different batch sizes. The numbers with the best performance are highlighted in red. 96
List of Figures

Figure 1.1 Given an input image (a). The three sub-problems of object recognition: (b) image classification, (c) object detection, and (d) semantic segmentation. ................................................................. 2

Figure 1.2 An illustration of image changes caused by different image acquisition effects. The original image (a) with defocus (b), random jerking (c), noise (d), and natural haze (e). .......................... 3

Figure 1.3 An illustration of image changes caused by different perspective angles. ................................................................. 4

Figure 1.4 An illustration of the variations in amorphous objects. The clouds are in the top row. The associated segmentation maps are in the bottom row. ................................................................. 5

Figure 1.5 An example of scale variation. The red boxes indicate objects near the camera. The green boxes indicate the objects far from the camera. ................................................................. 6

Figure 1.6 An illustration of variations in object size and occurrence, e.g., the “road” shows higher number in pixel quantity and instance occurrences than those of the “person”. .......................... 6

Figure 1.7 Given input images (a), the illustration of class segmentation (b) and instance segmentation (c). ................................................................. 7

Figure 2.1 The VGG16 architecture consists of thirteen convolutional layers followed by three fully-connected layers (including the softmax loss layer). ................................................................. 13

Figure 2.2 An illustration of the convolutional layer in a CNN architecture. The input image dimension is $1 \times 5 \times 5$, and the number of filters is $K = 1$, the convolution filter size is $F = 3$, the stride is $S = 2$, and the zero padding is $P = 1$. Therefore, use equation $(W - F + 2P) / S + 1$, the output size has spatial size $(5 - 3 + 2) / 2 + 1 = 3$. Note that a padding of $P = 1$ is applied to the input image, making the outer border of input image zero. .......................... 14
Figure 2.3 An illustration of the max pooling layer with $2 \times 2$ filter and 2 stride

Figure 2.4 Backpropagated gradients normalized histograms with hyperbolic tangent activation, with standard (top) vs Xavier (bottom) initialization. Top: 0-peak decreases for higher layers [36].

Figure 2.5 The Inception module in GooLeNet [108].

Figure 2.6 Residual learning: a building block.

Figure 2.7 A 5-layer dense block with a growth rate of $k = 4$.

Figure 2.8 An illustration of the FCN-8s [73].

Figure 2.9 An illustration of the SegNet [1].

Figure 2.10 An illustration of the PSPNet [126].

Figure 2.11 An illustration of the GCN [84].

Figure 2.12 An illustration of the U-Net [92].

Figure 4.1 Examples of semantic segmentation results on degraded images. From left to right, the six columns are degraded images, ground-truth segmentation, segmentation using the model trained on the clean images, segmentation using network fine-tuning on the degraded images, segmentation using network trained on both clean and degraded images (C&D), and segmentation using the proposed method trained on the degraded images.

Figure 4.2 An illustration of the proposed DGN architecture. The baseline network is FCN8s with VGG16 network backbone. The circled numbers denote the convolutional blocks.

Figure 4.3 Examples of semantic segmentation results on the degraded images. For each degradation effect, we select the degradation degrees of $d_1$, $d_3$, and $d_5$ for demonstration. The baseline segmentation network is FCN8s.
Figure 4.4  (a) The mIoUs (in percentage) of segmenting $d_5$ degree Gaussian blur images on PASCAL VOC 2012 dataset using: 1) Baseline networks fine-tuned using the degraded images; 2) Baseline networks trained using both clean and degraded images; 3) DGN trained using the degraded images; 4) DGN trained using paired clean and degraded images. (b) The dense-Gram loss $\mathcal{L}_{\text{Gram}}$ over 140,000 training iterations. The baseline segmentation network is FCN8s.

Figure 5.1  An illustration of the effectiveness of the proposed method, which improves the segmentation results of FCN-8s [73] by using proposed ISBEncoder, especially on small objects.

Figure 5.2  An overview of the proposed pipeline – segmentation network with the ISBEncoder. The ISBEncoder is capable of encoding the level of spatial adjacency between object-class pairs into the segmentation network.

Figure 5.3  An illustration of proposed ISBMetric of (a) ground-truth segmentation of an image, and (b-c) two sample segmentation results of the image. Below each segmentation are the corresponding ISBMetric matrix, ISBMetric accuracy and segmentation accuracy.

Figure 5.4  An illustration of the proposed ISBEncoder architecture.

Figure 5.5  Examples of semantic segmentation results on CamVID (left) and CityScapes (right) validation datasets. For visualization purpose, the ground-truth segmentation is superimposed to the input image, and the dashed rectangles are enlarged for highlighting improvements.

Figure 5.6  An illustration of the predicted ISBMetric $m_{\text{isb}}^{\text{pred}}$ and the ground truth ISBMetric $m_{\text{isb}}^{\text{gt}}$ of a sample image from CityScapes training dataset. The intensity of the colorbar denotes the value.

Figure 5.7  An illustration of the impact of the proposed ISBEncoder on the feature map after the “conv6”/the last convolutional layer in the PSPNet. The input image is from the CityScapes training dataset.
Figure 5.8  (a) The mIoU of large object classes, overall classes, and small object classes over 20k training iterations on CityScapes training dataset (in percentage). (b) The mean square error (MSE) loss of ISBEncoder over 20k training iterations. The baseline segmentation network is PSPNet.

Figure 6.1  An illustration of image-level and pixel-level imbalance in a widely used dataset. (a) Different classes in sample images in the CityScapes training dataset. (b) The proportion of the occurrence numbers of different classes at the image level. (c) The proportion of the sizes of different classes at the pixel level. For visualization purpose, the ground truth segmentation map is superimposed to the original image in (a), and the proportions are normalized in (b).

Figure 6.2  An illustration of the proposed FCN-based segmentation network with the proposed data balancing method. For demonstration purpose, the circled-checks and circled-crosses indicate the selected samples and dropped samples using the image-level data balancing method, respectively. The circled-bars indicate the samples that do not contain the class. The color lines with various thicknesses indicate weights of different classes.

Figure 6.3  Examples of semantic segmentation results on CityScapes dataset. For visualization purpose, the ground truth segmentation map is superimposed to the validation image and the dashed rectangles are enlarged for highlighting improvements.

Figure 6.4  Example results on segmenting brain tumor tissues: necrotic core (blue), oedema (green), non-enhancing (orange) and enhancing core (red). For visualization purpose, the dashed rectangles are used for highlighting improvements.

Figure 6.5  The segmentation performance additional gain (in percentage) over FCN-8s with WCE-based data balancing method using the CityScapes dataset. The baseline FCN-8s are trained with $\rho_t \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. 

xvi
CHAPTER 1

INTRODUCTION

Computer vision is an interdisciplinary scientific field that provides computing machines/computers with visual perception, i.e., endowing the machines the ability to see. In [2], visual perception is defined as the process of acquiring knowledge about environmental objects and events by extracting information from the light they emit or reflect (through any optical device such as an eye or a camera). This way, vision can be interpreted as the ability to model the perceivable world and achieve a high-level understanding of the scene. The “understanding” in this context means transforming low-level visual information into high-level descriptions of the world.

An essential task in computer vision is object recognition. Humans possess a remarkable ability to analyze an image and separate all the components present in an image, i.e., thoroughly parsing the image by knowing what (recognition) and where (localization) the objects are. Furthermore, humans can easily recognize objects that have never been seen before by observing a set of similar objects. However, it has been proved particularly challenging to build computing machines that can perform the object recognition effortlessly.

Based on the level of complexity, as shown in Fig. 1.1, object recognition can be roughly divided into three easy-to-hard sub-problems:

- Image classification: Identifying objects within an input image and providing image-level labels. The output labels are independent of object locations.

- Object detection: Identifying objects within an input image and generating a
Figure 1.1. Given an input image (a). The three sub-problems of object recognition: (b) image classification, (c) object detection, and (d) semantic segmentation.

- Semantic segmentation: Identifying objects within an input image and providing a label of a known semantic class to each pixel present in an image.

Among these three problems, semantic segmentation is arguably the most challenging problem in object recognition and it paves the way towards complete scene understanding. Many important applications are nourished from inferring knowledge from semantic segmentation, such as autonomous driving [25, 34, 18], human-machine interaction [81], medical imaging [39, 86, 96], computational photography [118], and image search engines [111]. In this thesis, we focus on studying semantic segmentation.

1.1 Challenges

Despite decades of research, semantic segmentation continues to be a very challenging problem and the current progress is still far from enough to handle many real-world applications. The main challenges come from different kinds of object and image variations in terms of appearance, spatial adjacency/relationship, image degradation, etc, which we discuss below.
Figure 1.2. An illustration of image changes caused by different image acquisition effects. The original image (a) with defocus (b), random jerking (c), noise (d), and natural haze (e).

1.1.1 Variations in Image Acquisition

We can expect there could be huge variations in the appearance of objects with different image acquisitions, caused by defocus, random jerking, camera noises, and environmental changes. Fig. 1.2 shows how the appearance of aeroplane changes under different image acquisitions. For the problem of semantic segmentation, the appearance of object, especially texture, shape, color are important for distinguishing one object from another. Different image acquisition can lead to different appearances of the same object, thus making semantic segmentation very difficult.

1.1.2 Variations in Perspectives

For the reason of that there exists information loss in projecting the real-world 3D objects to 2D images, a strong perspective usually leads to incorrect segmentation results. To tackle this problem, the training data must be diverse enough to cover the perspectives from different view angles, where objects could expressively represented.
Figure 1.3. An illustration of image changes caused by different perspective angles.

It is very time-consuming and expensive to obtain such diverse training sets. As shown in Fig. 1.3, we can see that the car is obviously different when viewing from different angles.

1.1.3 Variations in Amorphous Objects

The semantic objects can be categorized into two groups – objects with specific size and shapes (e.g., cars, person, dog), and objects with amorphous parts (e.g., cloud, sky, water). Different from the approaches, which utilize shape and size prior knowledge to segment rigid-body objects [3], as shown in Fig. 1.4, the approaches for amorphous object segmentation rely more on the texture of the objects [100]. Therefore, unsupervised clustering approaches become an attractive alternative, which divide the whole image into several groups depending on the similarity of the pixel intensity. However, the unsupervised clustering approaches tend to localize only few discrimina-
1.1.4 Variations in Object Scale

The object scale varies significantly on the 2D images caused by camera setting – an object appears to be larger (smaller) in size if it is closer to (farther away from) the camera. As demonstrated in Fig. 1.5, the person in the red boxes appears to be larger, whereas the person in the green boxes appears to be smaller. Such scale variation across object instances, and especially, the challenge of segmenting very small objects, stands out as one of the factors behind the difference in performance. The strategies of augmenting training set with resized images or using multi-resolution images have been proposed to alleviate this problem. However, these approaches are not effective by brutal-forcedly “remembering” the object features at pre-defined scales, not to
1.1.5 Variations in Object Size and Occurrence

As shown in Fig. 1.6, another challenge in semantic segmentation is the variations in object occurrence, where several classes (e.g., “road”, “sidewalk”, “cars”) show higher number in pixel quantity and instance occurrences than the other classes (e.g.,
Figure 1.7. Given input images (a), the illustration of class segmentation (b) and instance segmentation (c).

“person”, “traffic sign”, “pole”). When presented with complex imbalanced training data, the classifier would fail to properly represent the distributive characteristic of the data and may produce unfavorable segmentation for the minority classes [43]. The common approaches to tackle this problem is to re-sample the classes according to the training set. However, by only re-sampling to balance one class would affect the distribution of other classes in the training set.

1.2 Scope of the Research

With the high activity of advancement in the field of semantic segmentation, two important problems have been investigated by researchers: 1) Class segmentation; 2) Instance segmentation. In this section, we aim to describe those two problems, as well as the associated sub-problems in details.
1.2.1 Class Segmentation

Class segmentation is the most widely studied problem by far. The class segmentation, where each pixel is annotated using the identifier of a particular object (as shown in Fig. 1.7 (b)), are used to create segmentation map, such that each pixel is assigned a class label. Most of the recent semantic segmentation efforts are focused on recognizing and localizing thing classes, e.g., person, cat, dog in the PASCAL VOC 2012 challenge [26]. Such classes have specific sizes and shapes, and identifiable parts, e.g., a person has a head, two arms, and two legs. Recently, more attention has been given to the stuff classes, such as water, sky, and cloud, which are amorphous and have no distinct parts (e.g., a part of cloud is still cloud) [8, 26, 27, 18, 10]. One good reason of this phenomenon is that stuff often determines the type of a scene. For example, in a ski scene, the snow and sky are more essential components for the scene understanding than the ski-board and person. Stuff is also an essential component in reasoning objects – stuff encodes most of the scene layout and therefore provides strong constraints on the location of the things. For example, birds are usually flying in the sky. Recently, convolutional neural networks (CNNs) are widely adopted for semantic segmentation. State-of-the-art performance are achieved with the help of fully convolutional network (FCN) [73, 1, 92, 14, 15, 17]. However, current class segmentation approach still suffers from the common problems of image quality variations, global context embedding, and imbalanced training set, which many researchers are dedicated to [48, 40].

1.2.2 Instance Segmentation

Different from the class segmentation that makes predictions inferring labels for each pixel, instance segmentation includes identification of boundaries of the objects at the detailed pixel level. For example, as shown in Fig. 1.7 (c), there are three persons at certain locations, and it is expected to separate all three instances by highlighting
their pixels to blue, green, yellow, respectively. Instance segmentation is challenging because it requires the correct detection of all the objects in an image while also precisely segmenting each instance. It combines two components – the classical tasks of object detection, where the goal is to classify individual objects and localize each using a bounding box, and semantic segmentation, where the goal is to classify each pixel into a fixed set of classes without separating object instances. Recently, mask R-CNN [44] is proposed to tackle the problem of instance segmentation.

1.3 Proposed Approaches

In this thesis, we are interested in the problem of class segmentation. To tackle the challenge of variation in image acquisition, we propose a dense-Gram network to address the degradation effect induced bias. To tackle the challenges of variations in object perspective and scale, we propose an inter-class shared boundary based metric to encode the global context to the network. To tackle the challenge of variations in object size and occurrence, we propose a selective-weighting method to balance the training set.

1.3.1 Dense-Gram Network

We develop a dense-Gram Network (DGN) to tackle the problem of degraded image semantic segmentation. The proposed network consists of two identical networks – source and target networks. Both source and target networks are initialized using the model pre-trained on the clean image while only fixing the parameters of source network during training. The feature distribution in higher-layers is quantified using the Gram matrix [74]. Exploiting the capability of the Gram matrix in capturing the image textures, the Gram matrices from the source network can be considered as the image texture of the clean images, and the Gram matrices from the target network can be considered as the image texture of the degraded image. This way,
matching the Gram matrices between the source and target networks can 1) reduce the gap in feature distributions and 2) minimize the degradation effects induced bias, simultaneously. The detailed discussion of the proposed method is introduced in Chapter 4.

1.3.2 Inter-class Shared Boundary based Metric

We develop an inter-class shared boundary based metric (ISBMetric) to quantify the level of adjacency between each pair of object classes. Specifically, ISBMetric calculates the proportional length of the shared boundaries. For example, ISBMetric between object classes $A$ and $B$ is the proportion between the length of their shared boundaries and the perimeter of $A$ or $B$. In addition, by quantifying this ratio based level of spatial adjacency, the proposed ISBMetric is robust against object size induced biases, such that small objects can contribute more to the segmentation loss. We demonstrate that the enforcement of the ISBMetric can help improve the segmentation accuracy of small objects. We propose an ISBMetric based encoder for the purpose. In particular, the proposed ISBEncoder takes the prediction from the segmentation networks as the input and its output is guided by the ISBMetric matrix calculated using segmentation ground truth. The detailed discussion of the proposed method is introduced in Chapter 5.

1.3.3 Selective-weighting

We develop a selective-weighting method by exploiting data sampling and class weighting, to consider image- and pixel-level data balance simultaneously for FCN-based multi-class image segmentation networks. Furthermore, while many previous methods explored the data balance over the whole training set, we apply selective weighting over each batch of images in network training [55]. The detailed discussion of the proposed method is introduced in Chapter 6.
1.4 Structure of the Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we briefly overview the relevant knowledge and common networks used in this research. In Chapter 3, we review the related works of this thesis. In Chapters 4, 5, 6, we elaborate on degraded image semantic segmentation with dense-gram network, global context embedding with inter-class shared boundary based encoder, and data balancing with selective-weighting, respectively. A brief conclusion is discussed in Chapter 7.
Chapter 2

Background

In this chapter, an overview of relevant background knowledge, and algorithms used in this research is presented. The CNNs have shown outstanding performance in many computer vision problems and have become widely known standards. For this reason, we first present some details in the architecture of the CNNs. Secondly, we review some common CNNs backbones – VGG [101], GoogLeNet [108], ResNet [46] and DenseNet [49], which are currently being used as building blocks for many networks. Thirdly, we review some state-of-the-art networks for semantic segmentation tasks. Fourthly, we review some public semantic segmentation benchmark datasets.

2.1 Convolutional Neural Networks Basics

The CNNs consist of a sequence of layers, each transforming one volume of activations to another through linear or non-linear operators. There are three main distinct types of layers to build CNNs: convolutional layer, pooling layer, and fully-connected layer. An example of the CNN proposed by Simonyan and Zisserman [101] (VGG16) is shown in Fig. 2.1.

2.1.1 Convolutional Layers

The convolutional layer is the fundamental part of a CNN that computes the output of neurons that are connected to local regions in the input. The layer’s parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass,
The VGG16 architecture consists of thirteen convolutional layers followed by three fully-connected layers (including the softmax loss layer).

Each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input. Specifically, when dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume. The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.

Three parameters control the size of the output volume: the depth, stride and zero-padding. First, the depth corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edged, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a depth column. Second, the stride controls how depth columns around the spatial
Figure 2.2. An illustration of the convolutional layer in a CNN architecture. The input image dimension is $1 \times 5 \times 5$, and the number of filters is $K = 1$, the convolution filter size is $F = 3$, the stride is $S = 2$, and the zero padding is $P = 1$. Therefore, use equation $(W - F + 2P) / S + 1$, the output size has spatial size $(5 - 3 + 2)/2 + 1 = 3$. Note that a padding of $P = 1$ is applied to the input image, making the outer border of input image zero.

dimensions (width and height) are allocated. When the stride is 1, a new depth column of neurons is allocated to spatial positions only 1 spatial unit apart. This leads to heavily overlapping receptive fields between the columns, and also to large output volumes. Conversely, if higher strides are used then the receptive fields will overlap less and the resulting output volume will have smaller dimensions spatially. Third, the size of this zero-padding will allow us to control the spatial size of the output volumes. Specifically, we can compute the spatial size of the output volume as a function of the input volume size ($W$), the receptive field size of the Convolutional Layer neurons ($F$), the stride with which they are applied ($S$), and the amount of zero padding used ($P$) on the border. You can convince yourself that the correct formula for calculating how many neurons fit is given by $(W - F + 2P) / S + 1$. Fig. 2.2 illustrate the convolution operation in the convolutional layer, which is applied on a small region of the input data. Under the assumption that if one patch feature is useful to compute at some spatial position, then it should also be useful to compute at a different position, parameters are shared in convolutional layers to control the
number of free parameters.

2.1.2 Pooling Layers

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that once a feature has been found, its exact location isn’t as important as its rough location relative to other features. The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer in-between successive convolutional layers in a CNN architecture. The pooling operation provides a form of translation invariance.

The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size $2 \times 2$ applied with a stride of 2 down-samples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over four numbers. The depth dimension remains unchanged. In addition to max pooling, the pooling units can also perform other functions, such as average pooling and even $L_2$-Norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been found to work better in practice. Due to the aggressive reduction in the size of the representation, the current trend in the literature is towards using smaller filters [38] or discarding the pooling layer altogether [104].
Figure 2.3. An illustration of the max pooling layer with $2 \times 2$ filter and 2 stride

### 2.1.3 Fully-connected Layers

After several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully-connected layer have full connections to all neurons in the previous layer. The activations of a fully-connected layer can hence be computed using a matrix multiplication plus a bias offset. In CNNs, fully-connected layers encode the feature volume produced by convolutional layers to a feature vector, specific to a learning task. Usually, fully connected layers result in a main part of the total parameters in CNNs, which is likely to lead to overfitting. Therefore, some recent works [123, 122] remove the full connections between the final convolutional layer and followed fully-connected layer. As a result, the amount of parameter is largely reduced.

### 2.1.4 Xavier Initialization

Glorot et al. [36] proposed a weight initialization scheme such that the variance of the backpropagated gradients are roughly constant across layers, as shown in Fig. 2.4. Indeed, whereas the gradients have initially roughly the same magnitude, they diverge from each other (with larger gradients in the lower layers) as training progresses, especially with the standard initialization.

Note that this might be one of the advantages of the Xavier initialization, since
Figure 2.4. Backpropagated gradients normalized histograms with hyperbolic tangent activation, with standard (top) vs Xavier (bottom) initialization. Top: 0-peak decreases for higher layers [36].

having gradients of very different magnitudes at different layers may yield to ill-conditioning and slower training. The variances of backpropagated gradients are therefore treated as a key criteria in measuring the learning pace during learning/training.

2.1.5 MSRA/He Initialization

He et al. [45] proposed another initialization method regarding its proposed Parametric Rectified Linear Unit (PReLU). The motivation of proposing MSRA/He Initialization is by considering the the asymmetric distribution of the activation function.

2.1.6 Activation Functions

To further ensure the nonlinearity of the network, beside max pooling [47], nonlinear activation functions \(^1\) are proposed: logistic, tanh, softsign, rectified linear unit (ReLU), Leaky ReLU, Parametric ReLU. Note that, as pointed out in [36], the backpropagated gradients using sigmoid function can be easily saturated, which results in difficult updating/learning weights of the network. ReLU, on the other hand, aims

\(^1\)https://en.wikipedia.org/wiki/Activation_function
to preserve information about relative intensities as information travels through multiple layers of feature detectors [79]. Later, to avoid gradient introduced by ReLU, Leaky ReLU [115] is proposed. However, Leaky ReLU shows negligible impact on accuracy compared with ReLU. Then, to overcome the disadvantages of ReLU and leaky ReLU, PReLU [45] has demonstrated a new way in learning the slop of the negative part.

2.1.7 Dropout

Srivastava et al. [105] proposed a Dropout scheme to enable training the network in an ensemble fashion and make network robust against over-fitting. Specifically, the connections between units are randomly dropped out regarding a predefined dropping out rate. By default, the dropout rate is set to 0.5.

2.1.8 Batch Normalization

Ioffe and Szegedy [51] proposed a work towards reducing internal covariate shift. This work is motivated by the fact that the inputs to each layer are affected by the parameters of all preceding layers – so that small changes to the network parameters amplify as the network becomes deeper. In particular, the change in the distributions of layers’ inputs presents a problem because the layers need to continuously adapt to the new distribution. When the input distribution to a learning system changes, it is said to experience covariate shift [99]. Such covariate shifts are extended to sub-networks/layers, and make the network unstable during testing.

In recent networks, dropout is often replace by batch normalization for better performance.
2.1.9 Receptive Field

Luo et al. [76] studied the receptive fields of the kernel and proved that the distribution of impact in a receptive field distributes as a Gaussian. This leads to some intriguing findings, in particular that the effective area in the receptive field, which is called effective receptive field (ERF), only occupies a fraction of the theoretical receptive field, for the reason of that Gaussian distributions generally decay quickly from the center.

To reduce the Gaussian Damage in ERF, two approaches have been proposed by authors.

- New initialization: the weights at the center of the convolution kernel to have a smaller scale, and the weights on the outside to be larger. It will diffuses the concentration on the center out to the periphery.

- Architectural changes: instead of connecting each unit in a CNN to a local rectangular convolution window, we can sparsely connect each unit to a larger area in the lower layer using the same number of connections.

Also, we note that the effective receptive field in deep CNNs actually grows very slow, which indicates that a lot of local information is still preserved even after many convolution layers. If the ERF is smaller than the RF, this suggests that representations may retain position information, and also raises an interesting question concerning changes in the size of these fields during development.

2.2 Convolutional Neural Networks Backbones

State-of-the-art models for segmentation are based on adaptations of CNNs that had originally been designed for image classification. From network architecture perspective, the most straightforward way of improving the performance of deep neural networks is by increasing the size (both depth and width) of the network. The
Table 2.1. ImageNet top-1 and top-5 error rates. The networks are trained and validated using PyTorch.

<table>
<thead>
<tr>
<th>Network</th>
<th>Top-1 Error</th>
<th>Top-5 Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-11</td>
<td>30.98</td>
<td>11.37</td>
</tr>
<tr>
<td>VGG-13</td>
<td>30.07</td>
<td>10.75</td>
</tr>
<tr>
<td>VGG-16</td>
<td>28.41</td>
<td>9.62</td>
</tr>
<tr>
<td>VGG-19</td>
<td>27.62</td>
<td>9.12</td>
</tr>
<tr>
<td>VGG-11 BN</td>
<td>29.62</td>
<td>10.19</td>
</tr>
<tr>
<td>VGG-13 BN</td>
<td>28.45</td>
<td>9.63</td>
</tr>
<tr>
<td>VGG-16 BN</td>
<td>26.63</td>
<td>8.50</td>
</tr>
<tr>
<td>VGG-19 BN</td>
<td>25.76</td>
<td>8.15</td>
</tr>
<tr>
<td>Inception v3</td>
<td>22.55</td>
<td>6.44</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>30.24</td>
<td>10.92</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>26.70</td>
<td>8.58</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>23.85</td>
<td>7.13</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>22.63</td>
<td>6.44</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>21.69</td>
<td>5.94</td>
</tr>
<tr>
<td>Densenet-121</td>
<td>25.35</td>
<td>7.83</td>
</tr>
<tr>
<td>Densenet-161</td>
<td>22.35</td>
<td>6.20</td>
</tr>
<tr>
<td>Densenet-169</td>
<td>24.00</td>
<td>7.00</td>
</tr>
<tr>
<td>Densenet-201</td>
<td>22.80</td>
<td>6.43</td>
</tr>
</tbody>
</table>

object classification performances of the reviewed networks on ImageNet are shown in Table 2.1.

2.2.1 VGG

Simonyan and Zisserman [101] proposed VGG16 and VGG19, as shown in Fig. 2.1, for object classification. The overall network is characterized by its simplicity, using only $3 \times 3$ convolutional layers with $1 \times 1$ zero padding stacked on top of each other in increasing depth. The size is reduced by $2 \times 2$ max pooling. Two fully-connected layers, each of which consists of 4096 nodes, are then followed by a softmax classifier (cross-entropy loss). They first pre-train the smaller network and then use it as initialization for larger and deeper network. Fortunately, the network can be trained using Xaiver [36] or He [45] initialization. But still, there exist two major drawbacks of VGG:

- The training process is very slow.
- The overall architecture weights are large.
2.2.2 GoogLeNet / Inception

Szegedy et al. [108] proposed a 22 layers GoogLeNet using inception modules, as shown in Fig. 2.5. They pointed out that the goal of the inception module is to act as a multi-level feature extractor by using $1 \times 1$, $3 \times 3$, and $5 \times 5$ convolutions with the same module of the network – the output of these filters (top node, highlighted in green) are then stacked along the channel dimension and before being fed into the next layer in the network. The overall architecture weights is smaller than both VGG and ResNet (Inception v3 – 104MB). The architecture weights are further reduced in the extension work: Xception – 91MB.
2.2.3 ResNet

He et al. [46] proposed a residual learning scheme to circumvent the notorious problem of vanishing/exploding gradients [6, 36], and make training very deep (more than 1000 layers) network possible. As pointed out in [46], with the network depth increasing, accuracy gets saturated and then degrades rapidly, which hampers convergence from the beginning. To resolve this problem, as shown in Fig. 2.6, shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. The plain networks are trained with batch normalization, which ensures forward propagated signals to have non-zero variances. Also, batch normalization verifies that the back propagated gradients exhibit healthy norms. It is worth to note that He et al. trained the network by warming up a learning rate of \( lr = 0.01 \) until the training error is under 80% (about 400 iterations), and then tuning back to \( lr = 0.1 \) and continue training.

2.2.4 DenseNet

Huang et al. [49] proposed DenseNet for object detection using dense block. Different from traditional neural networks, DenseNet connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have
L connections, as shown in Fig. 2.7, the DenseNet consists of $\frac{L(L-1)}{2}$ direct connections. This network architecture makes intuitive sense in both the feedforward and backward setting. In the feed-forward setting, a task may benefit from being able to get low-level features activations in addition to high level feature activations. In the backward case, having all the layers connected allows the network to quickly send gradients to their respective places throughout the network effortlessly.

2.3 Semantic Segmentation Networks

Currently, the most successful state-of-the-art CNN based approaches for semantic segmentation derived from a common forerunner: the Fully Convolutional Network (FCN) [73]. The insight of that approach was to take advantage of existing CNNs as powerful visual models that can learn hierarchies of features. They transformed those existing and well-known classification models, e.g., VGG [101], GoogLeNet [108], and ResNet [46], into fully convolutional networks by replacing the fully connected layers with the convolutional layers to output the spatial maps instead of classification scores. Those maps are later up-sampled using transpose/fractionally-strided convolutions [73] to produce dense per-pixel labeled outputs. This work is considered a milestone as it demonstrated how CNNs can be trained end-to-end, efficiently learning how to make dense prediction for semantic segmentation with inputs of arbitrary sizes. In general terms, the FCN-based semantic segmentation network consists of 1) the encoder and 2) the decoder. The encoder usually is the part of the classification network, e.g., VGG, and with its fully connected layers removed. The encoder produces low-resolution images representations or feature maps. The decoder maps those low-resolution images or feature maps to pixel-wise predictions for semantic segmentation. In the following subsections, we will review the FCN and FCN-based semantic segmentations networks.
2.3.1 FCN

Long et al. [73] proposed fully convolutional neural network for segmentation, as shown in Fig. 2.8, which is the first successful attempt toward perform segmentation in a end-to-end fashion. A few key features of networks of this type are: 1) The features are merged from different stages/layers which vary in coarseness of the segmentation; 2) The upsampling of learned low resolution semantic feature maps is achieved by using deconvolutions/transpose-convolutions which are initialized with bilinear interpolation filters. 3) The knowledge transfer is adopted from object classification networks. The fully connected layers of classification networks are converted into fully convolutional layers. It produces a class presence heatmap in low resolution, which then is upsampled using bilinearly initialized deconvolutions and at each stage of upsampling further refined by fusing features from coarser but higher resolution feature maps from lower layers. The pooling layers are aimed to enable the succeeding block to extract more abstract, class-salient features from the pooled features.

On the other hand, the deconvolution layers allow the model to use every point in the small image to paint a square in the larger one. Unfortunately, deconvolution can easily have uneven overlap, putting more of the metaphorical paint in the same places than others [32]. In particular, deconvolution has uneven overlap when the ker-
nel size is not divisible by the stride. The uneven overlaps on the two axes multiply together, creating a characteristic checkerboard-like pattern \(^2\) of varying magnitudes. Thus, stride 1 deconvolution – which is often seen in successful models [73] – are quite effective at dampening checkerboard artifacts. Alternatively, dilated convolutions [120] can be used to address this problem. However, dilated convolutions are very expensive.

2.3.2 SegNet

Badrinarayanan et al. [1] proposed SegNet for semantic segmentation. The encoder of the SegNet is based on VGG-16. The decoder of SegNet is composed by a set of upsampling and convolutional layers which are at last followed by a softmax classifier to predict pixel-wise labels for an output which has the same resolution as the input image. As shown in Fig. 2.9, each upsampling layer in the decoder corresponds to a max-pooling one in the encoder. Those layers upsample feature maps using the max-pooling indices from their corresponding feature maps in the encoder phase. When the feature maps have been restored to the original resolution, they are fed to the softmax classifier to produce the final segmentation.

\(^2\)https://github.com/vdumoulin/conv_arithmetic
2.3.3 PSPNet

Zhao et al. [126] found that one of the issues in FCN-based networks is lack of suitable strategy to utilize global scene category clues. Inspired by spatial pyramid pooling, they proposed pyramid scene parsing network – PSPNet to consider both local and global clues in pixel prediction. As shown in Fig. 2.10, given an input image, the network first uses CNN to get the feature map of the last convolutional layer, then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information. Finally, the representation is fed into a convolutional layer to get the final per-pixel dense prediction.

2.3.4 GCN

Peng et al. [84] proposed Global Convolutional Network (GCN) for natural image semantic segmentation. As illustrated in Fig. 2.11, they follow two design principles: 1) From the localization view, the model structure should be fully convolutional to retain the localization performance and no fully-connected or global pooling layers should be used as these layers will discard the localization information; 2) From the classification view, large kernel size should be adopted in the network to enable densely connections between feature maps and per-pixel classifiers, which enhances
the capability to handle different transformation. By using the combination of $1 \times k + k \times 1$ and $k \times 1 + 1 \times k$ convolutions, the GCN enables densely connections within a large $k \times k$ region in the feature map.
2.3.5 U-Net

Ronneberger et al. [92] proposed a U-shape architecture to perform segmentation task in biomedical images. As shown in Fig. 2.12, the U-Net is an adaptation of FCN, which simply concatenates the encoder/down features maps to upsampled feature maps from the decoder/up at every stage to form a ladder like structure [88]. This architecture is beneficial since the decoder at each stage can learn back relevant features that are lost when pooled in the encoder. Due to the limitation in the quantity of biomedical images, excessive data augmentation is employed by applying elastic deformations to the available training images. Also, to further address the boundaries between cells, using Eq. 2.1, weight maps are calculated to compensate the different frequency of pixels from a certain class in the training data set, and to force the network to learn the small separation borders that are introduced between touching cells.

\[
\omega(x) = \omega_c(x) + \omega_0 \cdot \exp \left( -\frac{(d_1(x) + d_2(x))^2}{2\sigma^2} \right),
\]

where \( \omega_c : \Omega \rightarrow \mathbb{R} \) is the weight map to balance the class frequencies, \( d_1 : \Omega \rightarrow \mathbb{R} \) denotes the distance to the border of the nearest cell and \( d_2 : \Omega \rightarrow \mathbb{R} \) denotes the distance to the border of the second nearest cell. In U-Net experiment, \( \omega_0 = 10 \) and \( \sigma \approx 5 \) pixels.

2.4 Datasets

With the high activity of advancement in the field of semantic segmentation, it requires different public semantic segmentation benchmarks to fairly evaluate the effectiveness of each proposed method. The semantic segmentation benchmark datasets can be categorized into two major groups [10]: 1) Semantic objects/things segmentation datasets – segmenting objects with specific sizes/shapes (e.g., cars, person); 2) Semantic scene parsing datasets – segmenting the both objects with specific sizes/shapes
and stuffs with amorphous parts (e.g., cloud, sky).

**Semantic Object Segmentation Benchmark Datasets:** The semantic object segmentation datasets are conventional semantic segmentation benchmarks, which aim to segment the semantic objects (foreground) within the image. However, they usually lack the global context information (the relationship between different semantic objects), as the semantic objects are not spatially adjacent to each other.

- The PASCAL VOC 2012 dataset [26] is a natural object segmentation dataset which has been the benchmark challenge for segmentation over years. The dataset consists of 11,355 images with 8,498 training images and 2,857 validation images. In total, there are 21 semantic classes that are pixel-level annotated. Usually, each image in the PASCAL VOC 2012 dataset only contains one or two semantic objects, and the background is not labeled. As a result, the global spatial relationship between different semantic objects is very limited.

- The BRATS 2015 dataset [78] is a public benchmark for brain tumor segmentation. The training dataset consists of 220 cases of high grade glioma (HGG) and 54 cases of low grade glioma (LGG) patient scans. Each pixel is labelled as one of the five segmentation classes: necrotic core, oedema, non-enhancing, enhancing core and non-tumor. Four MRI modalities are available – FLAIR, T1, T1-contrast and T2. The data has been pre-processed with skull stripped and registered to a common space of dimension $240 \times 240 \times 155$ voxel. The segmentation performance is obtained via on-line submission. Since the size of the tumor is much smaller than that of the healthy tissue, the BRATS 2015 dataset is not balanced.

**Semantic Scene Parsing Benchmark Datasets:** To further understand the image, more attention has been given to stuff classes, such as grass and sky, which are amorphous in shapes and have no distinct parts (e.g. a piece of grass is still grass).
The semantic scene parsing benchmark datasets aim to segment both objects and amorphous parts. However, since the sizes and occurrences might vary significantly across different semantic objects, the datasets would not be well balanced.

- The CamVid dataset [8] is a road scene parsing benchmark dataset which is of current practical interest for various autonomous driving related problems. The dataset consists of 367 training images, and 100 validation images, and 233 testing images. In total, there are 11 semantic classes that are pixel-level annotated. Based on the object size [64], we denote sign symbol, pedestrian, pole, bicyclist as small-object classes. All the other 7 object classes are denoted as large-object classes. For the street scene images, the spatial relationship between different semantic objects amongst the dataset is relatively uniform.

- The CityScapes dataset [18] is a recently released dataset for semantic urban street scene understanding. The dataset consists of 5,000 finely pixel-level annotated images: 2,975 training images, 500 validate images, and 1,525 testing images. In total, there are 19 semantic classes. The resolution of the image is \(1,024 \times 2,048\). In addition, 20,000 coarsely annotated images are provided. In this research, we only use finely annotated images for training. Based on the object size [64], we denote pole, traffic light, traffic sign, person, rider, motorcycle and bicycle as small-object classes. All the other 12 object classes are denoted as large-object classes. All the other 12 object classes are denoted as large-object classes. For the street scene images, the spatial relationship between different semantic objects amongst the dataset is relatively uniform.

- The SUNRGBD dataset [103] is a challenging and large indoor scenes parsing benchmark dataset with both RGB images and the corresponding depth maps. The dataset consists of 5,285 training images and 5,050 testing images. In
total, there are 37 semantic classes that are pixel-level annotated. We only use the RGB images for training and testing. For the indoor scene images, the spatial relationship between different semantic objects amongst the dataset is not uniform.

Based on the features of each public dataset, we use the synthesized degraded images from four public datasets to evaluate the proposed method on the task of degraded image semantic segmentation: PASCAL VOC 2012, SUNRGBD, CamVid, and CityScapes. To validate the proposed method on the task of global context embedding, we use CamVid and CityScapes for evaluation, where the spatial adjacency between different semantic objects with each image is uniform across the dataset. To validate the proposed method on the task of imbalanced data training, we use two datasets: CityScapes and BRATS 2015 for evaluation, where the training data is not well balanced at both image and pixel levels.
Chapter 3

Literature Review

In this chapter, we first briefly overview the feature-based learning approaches. Then, we elaborate the related works for degraded image semantic segmentation, global context embedding, and data balancing, respectively.

3.1 An Overview of Feature-based Learning Approaches

The feature extraction approaches can be categorized into two groups: 1) Manually-crafted feature based approaches; 2) Feature learning based approaches. The chronicle of manually-crafted features can be traced back to 1970s and 1980s, where linear filters, e.g., Gaussian derivatives [3], Gabor wavelets [21], and Haar wavelets [107] are developed to extract edge/corner information of the image. In the late 1990s and early 2000s, the family of histogram-based features is developed, e.g., scale-invariant feature transformation (SIFT) [80], histogram of oriented gradients (HOG) [19] and speeded-up robust features (SURF) [4], to extract statistical information of the image. Due to the inter- and intra-class variations, it usually requires much more features to describe a complex objective. However, it is expensive in both time and labour to manually design features for the task, not to mention that different tasks/objects might require diverse expertise. The substantial burden of the manual feature crafting poses a potential obstacle in improving the performance. In the early 2000s, the performance improvement of semantic segmentation using manually-crafted features seems to reach its plateau.

Alternatively, in the 1990s, the feature learning-based approaches – feature rep-
representation learning – emerged. In this framework, instead of using manually-crafted features from an image, this group of approaches allow a machine to directly discover the features (representations) needed for the task. Not until the late 2000s, advances in hardware enables the computation-expensive machine learning algorithms, e.g., structured multi-layer perception – most often deep learning, renewed interest. In 2009, Nvidia was involved in what was called the “big bang” of deep learning, as convolutional neural networks (CNNs) were trained with Nvidia graphics processing units (GPUs). Specifically, GPUs are well-suited for the matrix/vector math involved in machine learning. GPUs speed up training algorithms by orders of magnitude, reducing running times from weeks to days. Furthermore, the performance of CNNs based approaches surpasses the performance of the conventional methods by a large margin in terms of accuracy and efficiency [101, 108, 46]. For the problem of semantic segmentation, it requires dense predictions for every pixel within the input image. The conventional “segmentation-by-classification” based approaches are considered to be less efficient in both training and testing phases. In 2015, Long et al. [73] achieved impressive results in PASCAL VOC 2012 [26] semantic segmentation challenge and dominate the field of semantic segmentation. The advantage of the FCN is to make the convolutional neural networks (CNNs) take arbitrary-sized images and output same-sized semantic segmentation maps. In the following, we discuss the related works for each challenge.

3.2 Related Works of Degraded Image Segmentation

For degraded image semantic segmentation, one naive strategy is to directly deploy the model, which is previously trained using clean images, to segment the degraded images. It is not surprising that the segmentation networks trained on clean images perform poorly on the degraded images. By design, CNN is a data-driven framework [101]. Differences in the representations of semantic objects between the clean
and degraded images would cause shifts in feature distributions [102], resulting in a decrease in segmentation performance [29]. Another strategy towards improving the segmentation performance of the degraded images is to first restore the degraded image to a clearer image such that human vision can better identify object and structure details present in the image. And then, we deploy the model, which is previously trained using the clean images, to segment the degraded images [102]. However, when the degradation degree is high, the image-restoration based pre-processing usually cannot completely remove those degradation effects, such that the restored images are different from their clean counterparts in both color and texture [83]. As demonstrated in Section 4.3.3, using the restored images for validation, most existing image-restoration methods only improve the degraded image semantic segmentation performance by a small margin. Recently, fine-tuning the network using the degraded images is a popular strategy for improving the segmentation performance [14, 70, 73]. However, fine-tuning based methods depend on the assumption that the segmentation networks can learn invariant representations that are transferable between the clean and degraded images [74]. The differences between the clean and degraded images pose a bottleneck to the feature transferability and hinder the segmentation network from further improvements. Also related to our work are the researches on domain adaptation [57, 68], style transfer [53], and knowledge distillation [91]. Kirkpatrick et al. [57] proposed an elastic weighting consolidation approach to remember the old tasks by selectively slowing down learning on the weights for the old tasks. Li et al. [68] proposed a Learning without Forgetting network, which uses only new classification-task data to train the network while preserving the original capabilities. The proposed DGN only considers the same tasks of semantic segmentation using the degraded images. Johnson et al. [53] proposed a perceptual loss to transfer the style of a fixed image to target images by matching the Gram matrices in a layer-wise manner. In contrast, the proposed DGN enhances the feature transferability by
matching the Gram matrices in a dense-interweaving manner. Romero et al. [91] proposed a Knowledge Distillation network, where knowledge is transferred from a large network to a smaller network for efficient deployment using the original input. Unlike the earlier approaches, the proposed DGN aims to find new network parameters for the same network structure with the goal of improving the semantic segmentation performance of the degraded images. Unlike the degraded images synthesized using clean images, in practice, the real degraded images are more difficult to collect and annotate, as well as quantifying their degradation levels. With limit-sized dataset, model overfitting could be a potential issue when using real images. In [129], Zhu et al. proposed an adversarial deep structured network to help address the issue. Yet, the proposed DGN uses the synthesized images to avoid the model overfitting issue.

3.3 Related Works of Global Context Embedding

In natural image semantic segmentation, various methods [14, 64] have been proposed to embed the global context into the model learning. One common strategy is to use multi-scale input images, such that the features of different-sized objects are captured during training. This strategy is usually implemented in CNNs, such that the network can learn both high-level large-scale and high-level small-scale features. Thus, it can reduce the object size induced biases. However, the training of the network requires data augmentation, not to mention the heavy time consumption in both training and testing. Another strategy to improve is to use context based post-processing, e.g., Markov Random Field (MRF) and fully connected Conditional Random Field (CRF). Most fully convolutional network (FCN) based methods exploit context information by constructing MRFs or CRFs as the post-processing stage to the overall network [14]. However, the post-processing stage is disconnected from the training of the network [14] and the network cannot adapt its weights based on the post-processing outputs [127]. Also related to this research are several scene parsing
approaches that partially address the small object segmentation problem. In [50], a Siamese network is proposed to learn the global context similarity between images. It tries to improve the segmentation performance of the small object classes by re-weighting the classes based on how often the classes appear in the dataset. In [66], a FoveaNet with CRFs is proposed to correct the distortion caused by the camera perspective projection. It tries to improve segmentation performance of the objects crowding around the vanishing point. In [98], a multi-stage feature is proposed for classification by bridging the connections between large objects and small objects with skip layers to increase the discrimination of small objects. In [97], multi-stage features are used to integrate global shape information with local distinctive information to learn the detectors. In [52], hinge loss is introduced to the CNNs resulting in a faster and more stable convergence with better performance. In [117], network is trained to generate multi-scale representation which enhances high-level small-scale features with multiple low-level feature layers. In [64], a perceptual generative adversarial network is proposed for small object detection, by minimizing representation difference between small objects and normal size objects. Several obstacle detection approaches [85, 87, 41] are proposed to detect potentially hazardous objects on the road. In [85], a stereo vision based method is proposed to detect obstacles on the road. In [41], a multi-stage MergeNet is proposed to detect obstacles, in which each stage tackles a different task. The output feature maps from each stage are later merged and used for the obstacle detection. Several RGB-D semantic segmentation approaches are also proposed. In [69], a context-aware receptive field (CaRF) is proposed to improve the segmentation performance. The CaRF provides better control over the relevant contextual information of the learned features, leading to a more focused domain which is easier to learn. In [112], a feature transformation network is proposed to improve the segmentation performance and bridge the convolutional networks and deconvolutional networks by discovering common features between RGB
images and depth maps. However, our work on small object segmentation is different from their works in task and usage, not to mention that the training of the models usually requires additional depth map/stereo images.

3.4 Related Works of Data Balancing

Although in some particular benchmark datasets (e.g., PASCAL VOC [27]), data balancing seems unnecessary [73], for most of the benchmark datasets (e.g., CityScapes and BRATS2015), data balancing is deemed crucial in improving the segmentation performance [14, 55]. Previous efforts aim to mitigate the data imbalance in semantic segmentation via lifting the importance of minority classes. Although many conventional data balancing methods can be applied to the problem of semantic segmentation, it cannot explicitly address image-level and pixel-level data balance simultaneously, not to mention that most of them are poorly suited for the FCN-based segmentation networks [114]. This is simply because: 1) Many conventional data balancing methods rely on hand-crafted features extracted from small data, which are inferior to deep features learned by CNN using big data; 2) Most of the conventional data balancing methods cannot perform batch-wise data balancing [23]. Existing methods for balancing the data in semantic segmentation can be mainly categorized into two directions: 1) Sampling based methods [28, 35]; 2) Cost-sensitive learning based methods [92, 14, 84]. The sampling based methods change the number of occurrences of classes at the image-level to compensate for imbalanced distribution between the majority and minority classes [28, 55]. This strategy is easy as it only needs to change the training dataset. Conventional methods include down-sampling majority classes, over-sampling minority classes or the combination of two. However, over-sampling can easily cause model over-fitting owing to repeatedly using the same samples [43]. Down-sampling may potentially throw away important information as it randomly discards samples from the majority classes until the effect of imbal-
ance is significant mitigated. In [28], Farabet et al. balance the data by sampling the dataset, such that the numbers of occurrences of classes are equal at the image level. In [55], Kamnitsas et al. apply data sampling to achieve image-level data balance in terms of the class label of the center pixel of training images. In [42], Hand et al. propose a data balancing method by sampling and weighing the classes in each training batch to improve the classification accuracy in the attribute classification tasks. However, these methods do not imply the aforementioned pixel-level data balance. The cost-sensitive learning based methods assign different weights to classes, such that the penalties for the majority and minority class samples are of similar costs [92, 14, 84]. One of the most popular cost-sensitive learning-based data balancing methods in FCN-based segmentation networks adopts WCE loss for data balancing [62]. It aims to approximately achieve pixel-level data balance for the whole training dataset [73, 92, 22]. In [1], Badrinarayanan et al. balance the classes by using median frequency balancing where the weight assigned to a class in the loss function is the ratio of the median of class frequency computed on the training dataset. In [14], Chen et al. also adopt median frequency balancing for data balancing. However, the weights are either estimated based on the training dataset or are provided by experts and are considered constant regardless of the variances between training batches, not to mention that the WCE loss does not pay enough attention to the image-level data balancing and cannot guarantee batch-wise data balancing.
Chapter 4
Degraded Image Segmentation with Dense-Gram Network

4.1 Motivation

In the real world, the degraded image semantic segmentation is a crucial enabler for safety-related applications, such as driving safety and precise navigation in autonomous driving and the highway navigation system [95]. The recent success of deep Convolutional Neural Networks (CNNs) has made remarkable progress in pixel-level semantic segmentation tasks [14, 70, 73]. Usually, the model trained using the clean images can be considered as the performance upper bound [110]. It has been shown that the performance of segmentation networks decreases under the degraded image quality [110]. As shown by an example in Fig. 4.1, the train is insufficiently segmented when directly employing the model pre-trained on the clean images. These errors are due to the drastic changes in the appearance of objects induced by the degradation. To our best knowledge, the degraded image semantic segmentation has not been systematically studied before. In the past decades, many approaches are developed for degradation removal [89], degraded image classification [75], degraded image detection [93], and general-purpose degraded image segmentation [12], but not much work on the semantic segmentation. In this research, we focus on developing a new approach towards degraded image semantic segmentation.

One straightforward strategy towards improving the performance of degraded image semantic segmentation is to remove the degradation effects by adding image-
Figure 4.1. Examples of semantic segmentation results on degraded images. From left to right, the six columns are degraded images, ground-truth segmentation, segmentation using the model trained on the clean images, segmentation using network fine-tuning on the degraded images, segmentation using network trained on both clean and degraded images (C&D), and segmentation using the proposed method trained on the degraded images.

restoration based pre-processing. However, when the degradation degree is high, the image-restoration based pre-processing usually cannot completely restore the degraded image to its clean counterpart and may introduce additional noise to the restored images [83]. Besides, in CNN based approaches, the image-restoration based pre-processing is not integrated into the segmentation network, which also affects the segmentation performance.

Most CNN based approaches seek to gain the robustness against the degradation effects by brutal-forcedly augmenting the training dataset – using both clean and degraded images for training [46, 101]. Specifically, a degraded image can be considered as the composition of its underlying clean image and certain additive degradation effects [20], where a degradation effect is treated as a type of image texture that does not necessarily depend on the location of semantic objects [30, 53].
Knowing that 1) the conventional network layers are not specifically tailored to effectively capture image texture [30, 31, 53] and 2) the supervised CNNs are task-driven frameworks, without the specific goal of capturing the image texture, the degradation effects cannot be properly addressed. Besides, using both clean and degraded images for training becomes increasingly time-consuming in training as more training images are added [40, 117].

With the emergence of image segmentation benchmarks [26, 103, 9, 18] and the high activity of advancement in the field of semantic segmentation, more and more models pre-trained on the clean images are made publicly available. Network fine-tuning on the existing models [73, 126, 14, 17] becomes a popular strategy towards improving the degraded image semantic segmentation performance. By design, the features learned in the higher/deeper layers of the network are semantic/task-related [37, 74]. We expect that the distribution of the higher-layer features learned using clean images should be similar to the distribution of the higher-layer features fine-tuned using degraded images [37, 68, 74, 119]. However, when fine-tuning the network using the degraded images, catastrophically forgetting the learned features of the clean images is inevitable [57]. This causes an increased gap in feature distributions of higher layers [74]. We observe that this gap in feature distributions poses a major obstacle in improving the segmentation performance of degraded images.

In this paper, we propose a novel approach to effectively reduce the gap and segment degraded images. The proposed network, as illustrated in Fig. 4.2, consists of two identical networks – source and target networks. Both source and target networks are initialized using the model pre-trained on the clean image while only fixing the parameters of source network during training. The feature distribution in higher-layers is quantified using the Gram matrix [74]. Exploiting the capability of the Gram matrix in capturing the image textures, the Gram matrices from the source network can be considered as the image texture of the clean images, and the
Gram matrices from the target network can be considered as the image texture of the degraded image. This way, matching the Gram matrices between the source and target networks can simultaneously 1) reduce the gap in feature distributions and 2) minimize the bias induced by degradation effects.

To enhance the transferability between the source and target networks, we match the Gram matrices in a dense-interweaving manner. Within the same convolutional block, the Gram matrix of the feature maps of a layer in one network is matched to Gram matrices of the feature maps of all layers of the same block in the other network. Because of this dense-interweaving manner, we refer to our approach as Dense-Gram Network (DGN). During deployment, we only use the trained target network to segment unseen degraded images, such that no extra time or cost is added to the segmentation network.

We evaluate the proposed DGN using synthetic degraded images generated based on four benchmark datasets: PASCAL VOC 2012 [26], SUNRGBD [103], CamVid [9], and CityScapes [18]. The proposed DGN is evaluated on different state-of-the-art segmentation networks and significantly outperforms the baselines when the degradation degree is high. To sum up, the main contributions of this paper are: 1) We systematically study the problem of the degraded image semantic segmentation. 2) We observe that gap between feature distribution learned using the clean images and the feature distribution learned using the degraded images poses a major obstacle in improving the segmentation performance of the degraded images. 3) We propose a novel DGN to segment degraded images and achieve substantially improved semantic segmentation performance on degraded images without adding extra time or cost during the deployment.
4.2 Method

In the following, we first give an overview of the proposed DGN in Section 4.2.1. Then, we elaborate the proposed dense-Gram loss and the proposed DGN training in Section 4.2.2. The analysis of the proposed DGN performance is discussed in Section 4.2.3.

4.2.1 Dense-Gram Network Overview

The architecture of the proposed DGN is based on the teacher-student networks [91], as illustrated in Fig. 4.2, where the source network provides “hints” for the training of the target network. The architecture of the source and target networks are identical and the parameters of both networks are initialized using the model trained on the clean images, while fixing the parameters of the source network during training. The total number of parameters in the proposed DGN is twice the number as the original network. During the network training, we only aim to train the target network. The
proposed DGN is trained in an end-to-end fashion. During the network testing, we only use the trained target network for evaluation. This way, we do not add extra time or cost during the deployment. In the following section, we elaborate on the proposed dense-Gram loss and the proposed DGN training.

4.2.2 Dense-Gram Loss & Network Training

Let \( \mathcal{D}_d = \{x_d^{(i)}, y^{(i)}\}_{i=1}^{n_d} \) denote the degraded image dataset with \( n_d \) labelled samples, where \( x_d \) and \( y \) denote the degraded images and the corresponding segmentation ground truth, respectively.

In CNNs, we define \( \psi(\cdot) \) as a composite function of following consecutive operations: batch normalization, followed by rectified linear unit and a \( 3 \times 3 \) convolution. Let \( f^{(l)}(x_d) = \psi^{(l)}(\psi^{(l-1)}(\cdots \psi^{(1)}(x_d))) \) denote the feature maps of the \( l^{th} \) layer. Let \( c^{(l)} \) denote the number of channels of the feature maps and let \( m^{(l)} \) denote the size (height times width) of the feature maps. Each element in the Gram matrix \( g \) is defined as:

\[
g^{(l)}_{i,j}(x_d) = \sum_{k=1}^{c^{(l)}} f^{(l)}_{i,k}(x_d) f^{(l)}_{j,k}(x_d),
\]

where \( i, j \in \{1, \cdots, c^{(l)}\} \) indicates the \( i^{th} \) and the \( j^{th} \) channels of the feature maps, and \( g^{(l)}_{i,j}(\cdot) \) is the value (at location \((i, j)\) of the Gram matrix \( g \)) of the inner product of the \( i^{th} \) and \( j^{th} \) vectorized feature maps of the respective \( i^{th} \) and the \( j^{th} \) channels in the \( l^{th} \) layer. Since the Gram matrix captures information about which features tend to activate together [53], the texture of the image can be well represented using the Gram matrix [30].

Let \( g^{(l)}_s(x_d) \) and \( g^{(l)}_t(x_d) \) denote the Gram matrix of the source and target networks at the \( l^{th} \) layer, respectively. The distance between the Gram matrices of the source and target networks is defined as follow:

\[
\delta^{(l,l)}_{\text{Gram}} = \frac{1}{4c^{(l)}^2m^{(l)}^2} \left\| g^{(l)}_s(x_d) - g^{(l)}_t(x_d) \right\|_2.
\]
To enhance the feature transferability between the source and target networks, within the same convolutional block, the Gram matrix of the feature maps of one layer in one network is matched to Gram matrices of the feature maps of all layers of the same block in the other network. Note that, within the same convolutional block, the dimensions of the feature maps generated from different layers are the same, i.e., for the $b^{th}$ block with $L^b$ layers, $c^{(l)} = c$, $m^{(l)} = m$, $\forall l \in \{1, \cdots, L^b\}$. The dense-Gram loss of the $b^{th}$ block is defined as:

$$
L_{\text{Gram}}^b = \frac{1}{4c^2m^2} \sum_{l=1}^{L^b} \sum_{l'=1}^{L^b} \| g_s^{(l)}(x_d) - g_t^{(l')} (x_d) \|_2^2.
$$

(4.3)

During the proposed DGN training, both source and target networks are initialized using the model trained on clean images, while fixing the parameters of the source network. As discussed in [54, 68], for teach-student network based designs, using the samples of new tasks and keeping the parameters of the source network (pre-trained using clean images) fixed could help the target network preserve similar classification performance of the source network. The network design of the proposed DGN is in-line with the “learning without forgetting” strategy discussed in [68] – requiring only degraded images for training while preserving the original segmentation performance on clean images. In this paper, the default DGN only uses the degraded images for training, when there is no ambiguity. In Section 4.3.2, we also report the segmentation performance of clean images using the DGN trained using degraded images.

For the semantic segmentation task, let $\theta_t(\cdot)$ and $\theta_s(\cdot)$ denote the target and source networks, respectively. Given the degraded image $x_d$ as the input of the proposed DGN, the prediction of the target network is $\theta_t(x_d)$. For the semantic segmentation task, we only use the segmentation loss of the target network for training. We follow [73] and adopt the sigmoid cross-entropy loss for optimization. The segmentation
The proposed DGN aims to train a target network $\theta_t(\cdot)$, which is guided by the source network $\theta_s(\cdot)$, to 1) reduce the gap between feature distribution learned using the clean and the feature distribution learned using degraded images, and 2) learn effective features for the target network in degraded image semantic segmentation task, i.e., $y = \theta_t(x_d)$. We first analyze the upper bound of segmentation performance of the proposed DGN.

Statistically, let probability distributions $p$ and $q$ characterize the distributions of the feature maps of the source and target networks, respectively. As proved in [67],
the gap between $p$ and $q$ can be measured using distance between the Gram matrices of the corresponding feature maps, i.e., $\delta_{\text{Gram}}$. The gap in feature distribution vanishes if and only if $p = q$.

Let $\epsilon_d(\theta_t) = \Pr_{(x_d,y) \sim q} [\theta_t(x_d) \neq y]$ and $\epsilon_d(\theta_s) = \Pr_{(x_d,y) \sim p} [\theta_s(x_d) \neq y]$ be the risks of the target and source network when using the degraded images for training, respectively. Noted by the proof in [5, 74], the target risk can be bounded by

$$\epsilon_d(\theta_t) \leq \epsilon_d(\theta_s) + 2\delta_{\text{Gram}} + C,$$

(4.6)

where $C$ is a constant for the risk of an ideal null hypothesis, i.e., $p = q$, for both feature distributions and the complexity of the hypothesis space.

Let $\epsilon_c(\theta'_s) = \Pr_{(x_c,y) \sim p'} [\theta'_s(x_c) \neq y]$ be the risk of the source network when using the clean images for training, where $x_c$ denotes the clean images. We can expect the risk of source network to be bounded by $\epsilon_c(\theta'_s) \leq \epsilon_d(\theta_s)$. Since the source network is fixed during training, the feature distributions of the source network remain unchanged, i.e., $p = p'$, such that $\epsilon_c(\theta'_s) = \epsilon_c(\theta_s)$. Based on Eq. (4.6), the risk of the target network can be bounded by

$$\epsilon_d(\theta_t) \leq 2\delta_{\text{Gram}} + 2\epsilon_d(\theta_s) - \epsilon_c(\theta_s) + C$$

$$= 2\delta_{\text{Gram}} + C',$$

(4.7)

where $C' = 2\epsilon_d(\theta_s) - \epsilon_c(\theta_s) + C$. When using paired clean and degraded images as the input of the respective source and target networks, i.e., $\epsilon_d(\theta_s) = \epsilon_c(\theta_s)$, the upper bound of the target risk $\epsilon_d(\theta_t)$ can be further reduced to $\epsilon_d(\theta_t) \leq 2\delta_{\text{Gram}} + \epsilon_c(\theta_s) + C$.

As can be seen, this $2\delta_{\text{Gram}} + \epsilon_c(\theta_s) + C$ is the performance upper bound of what our proposed DGN can achieve. In the later experiment, we report the performance upper bound of the proposed DGN in Section 4.3.2.

Secondly, we analyze the rationale of the minimization of the dense-Gram loss. During network training, the proposed DGN performs two tasks: 1) Semantic segmentation; 2) Densely-interweaving Gram matrices matching. The proposed DGN
Figure 4.3. Examples of semantic segmentation results on the degraded images. For each degradation effect, we select the degradation degrees of $d_1$, $d_3$, and $d_5$ for demonstration. The baseline segmentation network is FCN8s.

seeks to learn optimal network parameters by jointly minimizing the segmentation loss and the dense-Gram loss, such that the target risk $\epsilon_d(\theta_t)$ can be reduced. By design, minimizing the segmentation loss can effectively minimize the target risk $\epsilon_d(\theta_t)$. Furthermore, based on Eq. (4.7), it is $\delta_{\text{Gram}}$ and $C'$ that affect the upper bound of the target risk. Since the source network is fixed during training, the source risk $\epsilon_d(\theta_s)$ becomes a constant. Therefore, by minimizing the dense-Gram loss, the upper bound of the target risk in the proposed DGN can be further decreased, which leads to an improved segmentation performance.
Table 4.1. The mIoUs (in percentage) of segmenting degraded images using: 1) Baseline networks fine-tuning using the degraded images (+fine-tune); 2) Baseline networks trained using both clean and degraded images (+C&d); 3) DGN trained using the degraded images (+DGN); 4) DGN trained using paired clean and degraded images (+DGN+C&D). “Clean” denotes the mIoUs on the clean images. The five degradation degrees are denoted using d1, d2, d3, d4, and d5, respectively. The numbers with the better and best performance are highlighted in blue and red, respectively.

### Pascal VOC 2012

<table>
<thead>
<tr>
<th>Degradation</th>
<th>Clean</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>86.6</td>
<td>86.4</td>
<td>86.7</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>86.6</td>
<td>86.4</td>
<td>86.7</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td>FPN+C&amp;D</td>
<td>86.6</td>
<td>86.4</td>
<td>86.7</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td>FPN+DGN</td>
<td>86.6</td>
<td>86.4</td>
<td>86.7</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td>FPN+DGN+C&amp;D</td>
<td>86.6</td>
<td>86.4</td>
<td>86.7</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
</tbody>
</table>

### Cityscape

<table>
<thead>
<tr>
<th>Degradation</th>
<th>Clean</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+C&amp;D</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+DGN</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+DGN+C&amp;D</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
</tbody>
</table>

### CanCan

<table>
<thead>
<tr>
<th>Degradation</th>
<th>Clean</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+C&amp;D</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+DGN</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+DGN+C&amp;D</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
</tbody>
</table>

### SGGdet

<table>
<thead>
<tr>
<th>Degradation</th>
<th>Clean</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
<td>mIoU</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+slow-time</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+C&amp;D</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+DGN</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>FPN+DGN+C&amp;D</td>
<td>87.0</td>
<td>86.8</td>
<td>87.1</td>
<td>87.1</td>
<td>86.9</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+slow-time</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>ResNet+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
<tr>
<td>DLA+DGN+C&amp;D</td>
<td>88.5</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
<td>88.3</td>
</tr>
</tbody>
</table>
Table 4.2. The mIoUs (in percentage) of segmenting clean images using 1) fine-tuned baseline; 2) trained DGN; 3) trained DGN+C&D. The five degradation degrees are denoted using $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>Degradation</th>
<th>FCN8s+Fine-tune</th>
<th>FCN8s+DGN</th>
<th>FCN8s+DGN+C&amp;D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Blur</td>
<td>66.9 64.2 60.4 56.0 52.4</td>
<td>67.2 64.1 61.2 58.9 55.9</td>
<td>67.2 65.5 64.7 57.9 55.8</td>
</tr>
<tr>
<td>Motion Blur</td>
<td>66.2 64.3 60.4 56.0 52.4</td>
<td>67.2 65.4 64.3 57.9 55.8</td>
<td>67.2 65.5 64.7 57.9 55.8</td>
</tr>
<tr>
<td>S &amp; P Noise</td>
<td>66.2 64.3 60.4 56.0 52.4</td>
<td>67.2 65.4 64.3 57.9 55.8</td>
<td>67.2 65.5 64.7 57.9 55.8</td>
</tr>
<tr>
<td>Haze</td>
<td>66.2 64.3 60.4 56.0 52.4</td>
<td>67.2 65.4 64.3 57.9 55.8</td>
<td>67.2 65.5 64.7 57.9 55.8</td>
</tr>
</tbody>
</table>

4.3 Experiments

4.3.1 Implementation Details

The proposed pipeline is conducted using PyTorch 1, MatConvNet 2, and Tensorflow 3 implementations with Intel Core i7 6700K and with Nvidia 1080Ti GPUs with mini-batch Stochastic Gradient Descent (SGD). The segmentation network is trained using SGD with momentum of 0.9, weight decay of 0.0001 and adaptive learning rates. The mini-batch size is 1. When training networks using both clean and degraded images for baseline comparison methods, we follow the same practices reported in the original papers. When fine-tuning the networks, we follow the standard procedure for the network training [68], where smaller learning rates are adopted ($10^{-1}$ times the original rate). The initial learning rates for the FCN-8s [73], DeepLab v2 [14], DeepLab v3 Plus [17], RefineNet [70], EncNet [124], PSPNet [126], and DLA [121] are $1 \times 10^{-11}$, $1 \times 10^{-5}$, $7 \times 10^{-5}$, $5 \times 10^{-5}$, $1 \times 10^{-4}$, $1 \times 10^{-4}$, $1 \times 10^{-3}$, respectively. We use the “poly' learning rate policy [14], where the initial learning rate is multiplied by $\left(1 - \frac{\text{iter}}{\text{max iter}}\right)^{\text{power}}$ with $\text{power} = 0.9$ [126]. As suggested by [1], the number of training iterations is 140,000 for all experiments.

1https://github.com/pytorch

2http://www.vlfeat.org/matconvnet/

3https://www.tensorflow.org/
The degradation effects include: Gaussian blur, linear motion blur, salt & pepper noise, and haze. To better demonstrate the effectiveness of the proposed DGN, we evaluate each degradation effect using five degradation degrees $d$:

- The degree of the Gaussian blur is quantified by the standard deviation of the Gaussian kernel, $d \in \{1, 2, 3, 4, 5\}$.
- The degree of the linear motion blur is quantified by the motion length, $d \in \{5, 10, 15, 20, 25\}$.
- The degree of salt & pepper noise is quantified by the noise density, $d \in \{0.02, 0.04, 0.06, 0.08, 0.10\}$.
- The degree of haze is quantified by the scattering coefficient of atmosphere, $d \in \{1.5, 2.0, 2.5, 3.0, 3.5\}$.

For fair comparison, the training data is the same for each type of degraded images without adding any extra data. The only exceptions are made when 1) training the baseline networks using both clean and degraded images, and 2) training the DGN using the paired clean and degraded images. In this research, the approaches using both clean and degraded images are denoted using “C&D" as postfix, where there is no ambiguity.

4.3.2 COMPARING TO BASELINE SEGMENTATION NETWORKS

To demonstrate the effectiveness of the proposed DGN, we train and evaluate the proposed DGN in four datasets using the baseline segmentation networks with published pre-trained models. For PASCAL VOC 2012 dataset, five baseline networks – FCN8s, DeepLab v2, RefineNet, EncNet, and DeepLab v3 Plus – are evaluated. For SUNRGBD dataset, three baseline networks – FCN8s, DeepLab v2, and RefineNet – are evaluated. For CamVid dataset, three baseline networks – FCN8s, DeepLab v2,
Figure 4.4. (a) The mIoUs (in percentage) of segmenting $d_5$ degree Gaussian blur images on PASCAL VOC 2012 dataset using: 1) Baseline networks fine-tuned using the degraded images; 2) Baseline networks trained using both clean and degraded images; 3) DGN trained using the degraded images; 4) DGN trained using paired clean and degraded images. (b) The dense-Gram loss $\mathcal{L}_{\text{Gram}}$ over 140,000 training iterations. The baseline segmentation network is FCN8s.

Table 4.3. The mIoUs (in percentage) of segmenting degraded images using: 1) Pre-trained model tested using the restored images; 2) Baseline network fine-tuned using the restored images (*+fine-tune); 3) DGN trained using the degraded images (DGN); 4) DGN trained using the restored images (*+DGN). The five degradation degrees are denoted using $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively. The numbers with better and the best performance are highlighted in blue and red, respectively.
and DLA – are evaluated. For CityScapes dataset, four baseline networks – FCN8s, PSPNet, DLA, DeepLab v3 Plus – are evaluated. The quantitative experiment results are shown in Table 4.1.

Firstly, we compare the proposed DGN to the fine-tuning based counterparts. The proposed DGN achieves substantial improvements when the degradation degree is high, e.g., \(d_5\). Specifically, using the PASCAL VOC 2012 testing dataset for evaluation, it shows averagely 3.2\%, 3.4\%, 3.5\%, 3.1\%, 3.7\% improvements for FCN8s, DeepLab v2, RefineNet, EncNet, and DeepLab v3 Plus. Using the SUNRGBD testing dataset for evaluation, it shows averagely 1.9\%, 2.7\%, and 3.1\% improvements for FCN8s, DeepLab v2, and RefineNet. Using the CamVid testing dataset for evaluation, it shows averagely 3.4\%, 3.6\%, and 3.7\% improvements for FCN8s, DeepLab v2, and DLA. Using the CityScapes testing dataset for evaluation, it shows averagely 3.7\%, 4.0\%, 3.5\%, 3.3\% improvements for FCN8s, PSPNet, DLA, and DeepLab v3 Plus.

Secondly, we compare the proposed DGN to the baseline networks fine-tuned using both clean and degraded images. Note that the proposed DGN only use the same degraded images for training without adding any extra data. Using the PASCAL VOC 2012 testing dataset for evaluation, it shows averagely 1.2\%, 1.7\%, 1.6\%, 1.7\%, 1.7\% improvements for FCN8s, DeepLab v2, RefineNet, EncNet, and DeepLab v3 Plus. Using the SUNRGBD testing dataset for evaluation, it shows averagely 1.5\%, 1.6\%, and 1.5\% improvements for FCN8s, DeepLab v2, and RefineNet. Using the CamVid testing dataset for evaluation, it shows averagely 1.8\%, 1.8\%, and 1.9\% improvements for FCN8s, DeepLab v2, and DLA. Using the CityScapes testing dataset for evaluation, it shows averagely 1.9\%, 2.0\%, 1.7\%, 1.9\% improvements for FCN8s, PSPNet, DLA, and DeepLab v3 Plus.

Thirdly, as discussed in Section 4.2.3, we conduct the experiments to test the upper bound performance of the proposed DGN. We use paired clean and degraded
images as the respective inputs of the source and target networks and compare the segmentation performance to the proposed DGN. Using the *PASCAL VOC 2012* testing dataset for evaluation, it shows averagely 0.9%, 0.8%, 0.7%, 0.8%, 0.8% additional improvements for FCN8s, DeepLab v2, RefineNet, EncNet, and DeepLab v3 Plus. Using the *SUNRGBD* testing dataset for evaluation, it shows averagely 0.9%, 0.8%, and 1.0% additional improvements for FCN8s, DeepLab v2, and RefineNet. Using the *CamVid* testing dataset for evaluation, it shows averagely 0.8%, 1.0%, and 0.8% additional improvements for FCN8s, DeepLab v2, and DLA. Using the *CityScapes* testing dataset for evaluation, it shows averagely 0.8%, 0.9%, 1.0%, 0.8% additional improvements for FCN8s, PSPNet, DLA, and DeepLab v3 Plus.

We conduct an additional experiment to evaluate the segmentation performance of the clean images using fine-tuned baseline, trained DGN, and trained DGN+C&D. The experimental results are reported in Table 4.2. In comparison to the fine-tuned baseline, the proposed DGN+C&D constantly achieves the best performance. In comparison to the baseline pre-trained and evaluated using the clean images, the DGN trained using the degraded images only decrease the performance by a small margin.

To better understand the relationship between the segmentation performance and the dense-Gram loss, as shown in Fig. 4.4, we provide a sample of training curves on *PASCAL VOC 2012* $d_5$ degree Gaussian blur images. Note that we follow the same network design as the proposed DGN to only calculate the dense-Gram losses for the fine-tuning based and C&D based approaches. Using the dense-Gram loss for quantification, we observe that, when fine-tuning the network using the degraded images, the gap in the distributions of features learned using the clean and degraded images first drops but then increases along with the training iterations. The C&D based approach reduces the gap, but not very significant. This pattern of the increased gap is similar to the findings discussed in [37, 68]. On the other hand, the proposed DGN
Table 4.4. The mIoUs (in percentage) of segmenting degraded images using: 1) DGN; 2) Direct feature maps matching (DGN-MSE); 3) Linear kernel (DGN-Linear); 4) Gaussian kernels (DGN-Gaussian); 5) Layer-wise Gram matrices matching (DGN-Layerwise). For each degradation effect, the degradation degree is increased from left to right. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>Degradation Effect</th>
<th>PASCAL VOC 2012</th>
<th>SUNRGBD</th>
<th>CamVid</th>
<th>CityScapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Blur</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGN</td>
<td>67.2 64.8 64.3</td>
<td>63.5</td>
<td>63.6</td>
<td>63.5</td>
</tr>
<tr>
<td>DGN-MSE</td>
<td>67.0 64.8 64.1</td>
<td>63.7</td>
<td>63.6</td>
<td>63.5</td>
</tr>
<tr>
<td>DGN-Linear</td>
<td>67.5 64.8 64.3</td>
<td>63.4</td>
<td>63.6</td>
<td>63.5</td>
</tr>
<tr>
<td>DGN-Gaussian</td>
<td>67.9 65.9 64.0</td>
<td>63.1</td>
<td>63.7</td>
<td>63.6</td>
</tr>
<tr>
<td>DGN-Layerwise</td>
<td>68.0 65.9 64.9</td>
<td>63.2</td>
<td>63.7</td>
<td>63.6</td>
</tr>
<tr>
<td>Motion Blur</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGN</td>
<td>64.5 65.2 51.4</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-MSE</td>
<td>64.5 65.2 51.4</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-Linear</td>
<td>64.5 65.2 51.4</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-Gaussian</td>
<td>64.5 65.2 51.4</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-Layerwise</td>
<td>64.5 65.2 51.4</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>S &amp; P Noise</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGN</td>
<td>66.1 66.5 60.0</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-MSE</td>
<td>66.1 65.2 60.0</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-Linear</td>
<td>66.1 65.2 60.0</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-Gaussian</td>
<td>66.1 65.2 60.0</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>DGN-Layerwise</td>
<td>66.1 65.2 60.0</td>
<td>64.4</td>
<td>64.4</td>
<td>64.4</td>
</tr>
<tr>
<td>Base</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DGN</td>
<td>68.5 68.9 65.4</td>
<td>63.6</td>
<td>63.6</td>
<td>63.6</td>
</tr>
<tr>
<td>DGN-MSE</td>
<td>68.5 68.9 65.4</td>
<td>63.6</td>
<td>63.6</td>
<td>63.6</td>
</tr>
<tr>
<td>DGN-Linear</td>
<td>68.5 68.9 65.4</td>
<td>63.6</td>
<td>63.6</td>
<td>63.6</td>
</tr>
<tr>
<td>DGN-Gaussian</td>
<td>68.5 68.9 65.4</td>
<td>63.6</td>
<td>63.6</td>
<td>63.6</td>
</tr>
<tr>
<td>DGN-Layerwise</td>
<td>68.5 68.9 65.4</td>
<td>63.6</td>
<td>63.6</td>
<td>63.6</td>
</tr>
</tbody>
</table>

continuously decreases the gap and further improves the segmentation performance.

As can be seen, in comparison to the fine-tuning based and C&D based strategies, the proposed DGN is more effective in degraded image semantic segmentation. As shown in Fig. 4.3, we provide sample qualitative segmentation results for demonstration. In comparison to the segmentation results based on the network fine-tuning, the proposed DGN obtains visually better results, especially when the degradation degree is high, which can accurately classify the components when using network fine-tuning and render more accurate segmentation results.

4.3.3 Impact of Image Restoration Based Pre-processing

We conduct experiments to evaluate whether the image restoration based pre-processing could help the degraded image segmentation. Note that we select FCN8s as the baseline network for validation because it is fast to train and is well-studied by the community. The Gaussian blurred images are deblurred using the conventional deconvblind 4, linear motion blurred images are deblurred using DeblurGAN [60], images...

Table 4.5. The mIoUs (in percentage) of segmenting degraded images using different \( \lambda \) in Eq. (4.5), default \( \lambda = 1 \times 10^{-1} \). The five degradation degrees are denoted using \( d_1, d_2, d_3, d_4, \) and \( d_5 \), respectively. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Degradation</th>
<th>( \lambda = 10^{-1} )</th>
<th>( \lambda = 1 \times 10^{-2} )</th>
<th>( \lambda = 1 \times 10^{-3} )</th>
<th>( \lambda = 3 \times 10^{-4} )</th>
<th>( \lambda = 5 \times 10^{-5} )</th>
<th>( \lambda = 10^{-5} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamVid</td>
<td>Gaussian Blur</td>
<td>66.8, 66.3, 65.8, 65.2, 66.7, 65.7, 64.9</td>
<td>66.3, 65.5, 64.6, 63.4, 61.7</td>
<td>66.9, 66.4, 66.0, 65.8, 65.0, 63.8, 62.3</td>
<td>64.9, 64.7, 64.4, 63.8, 62.5, 61.3, 60.4</td>
<td>62.2, 61.4, 60.9, 60.2, 58.5</td>
<td>60.0, 59.7, 59.2, 58.9, 58.6, 58.2, 57.9</td>
</tr>
<tr>
<td>CityScapes</td>
<td>Gaussian Blur</td>
<td>65.4, 59.7, 54.7, 52.2, 48.1</td>
<td>66.8, 66.3, 66.0, 67.1, 64.6, 62.1, 57.3</td>
<td>65.3, 59.5, 54.6, 52.1, 48.1, 47.5, 47.0</td>
<td>64.8, 64.4, 64.2, 63.8, 63.2, 62.8, 62.5</td>
<td>63.9, 63.2, 61.1, 60.1, 58.9</td>
<td>62.0, 62.6, 61.8, 62.0, 60.4</td>
</tr>
<tr>
<td>SUNRGBD</td>
<td>Motion Blur</td>
<td>64.9, 58.5, 52.2, 49.9, 47.5</td>
<td>65.4, 59.7, 54.7, 52.2, 48.1, 47.5, 47.0</td>
<td>65.3, 59.6, 54.6, 52.1, 48.1, 47.5, 47.0</td>
<td>64.8, 64.4, 63.8, 63.2, 62.5</td>
<td>64.1, 64.4, 63.8, 62.5</td>
<td>64.4, 64.3, 63.9, 63.3, 62.6</td>
</tr>
<tr>
<td>PASCAL VOC 2012</td>
<td>S &amp; P Noise</td>
<td>66.4, 65.5, 64.0, 63.1, 62.7, 61.5, 60.8</td>
<td>66.4, 65.5, 64.0, 63.1, 62.7, 61.5, 60.8</td>
<td>65.8, 65.0, 64.4, 63.4, 62.7, 61.7, 60.9</td>
<td>65.3, 65.0, 64.4, 63.4, 62.7, 61.7, 60.9</td>
<td>64.8, 65.0, 64.4, 63.4, 62.7, 61.7, 60.9</td>
<td>64.3, 65.0, 64.4, 63.4, 62.7, 61.7, 60.9</td>
</tr>
<tr>
<td>CityScapes</td>
<td>Haze</td>
<td>66.8, 66.3, 66.3, 65.9, 65.5, 65.1, 64.8</td>
<td>66.8, 66.3, 66.3, 65.9, 65.1, 64.8, 64.5</td>
<td>66.8, 66.3, 66.3, 65.9, 65.1, 64.8, 64.5</td>
<td>66.8, 66.3, 66.3, 65.9, 65.1, 64.8, 64.5</td>
<td>66.8, 66.3, 66.3, 65.9, 65.1, 64.8, 64.5</td>
<td>66.8, 66.3, 66.3, 65.9, 65.1, 64.8, 64.5</td>
</tr>
</tbody>
</table>

with salt & pepper noise are resorted using median filter \(^5\), hazy images are dehazed using: CAP dehaze [128], DehazeNet [11], and DCPDN [125]. The experiments are conducted in three respects: 1) Test the restored images using the model trained on the clean images; 2) Fine-tune the network using the restored images; 3) Train the proposed DGN using the restored images.

The quantitative results are reported in Table 4.3. It is not surprising to observe a relative poor segmentation performance when directly test the restored images using the model pre-trained on the clean image. This is simply because that the image restoration based pre-processing usually cannot completely restore the degraded images to their clean counterparts. Not to mention that the image restoration based pre-processing can potentially modify both texture and color information of the image and could introduce additional noise to the restored images and result in a relative poor segmentation performance.

One exception is observed when using the median filter to remove the salt & pepper noise on CamVid and CityScapes datasets. The segmentation performance

\(^5\)https://www.mathworks.com/help/images/ref/medfilt2.html

56
Table 4.6. The mIoUs (in percentage) of segmenting degraded images when the dense-Gram matching begins at the 2nd, 3rd, 4th, 5th, and 6th convolutional blocks. For each degradation effect, the degradation degree is increased from left to right. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th></th>
<th>Gaussian Blur</th>
<th>Motion Blur</th>
<th>S &amp; P Noise</th>
<th>Haze</th>
</tr>
</thead>
<tbody>
<tr>
<td>ρ = 0</td>
<td>65.6 63.8 62.2 60.5 60.4</td>
<td>66.4 65.1 59.2 51.3 48.4</td>
<td>64.9 59.8 54.5 48.9</td>
<td>66.8 66.2 65.0 62.1 60.3</td>
</tr>
<tr>
<td>ρ = 1</td>
<td>65.3 63.6 62.0 60.3 60.2</td>
<td>66.3 64.9 59.1 51.1 48.2</td>
<td>64.7 59.5 54.0 48.3</td>
<td>66.7 66.1 64.9 61.7 59.7</td>
</tr>
<tr>
<td>ρ = 2</td>
<td>65.0 63.3 61.7 60.0 59.9</td>
<td>66.0 64.6 59.0 50.9 48.0</td>
<td>64.4 58.9 53.9 47.9</td>
<td>66.4 66.0 64.8 61.6 60.0</td>
</tr>
<tr>
<td>ρ = 3</td>
<td>64.7 63.0 61.5 60.8 60.7</td>
<td>65.8 64.4 58.9 51.0 48.1</td>
<td>64.1 58.6 53.6 47.6</td>
<td>66.1 65.9 64.7 61.4 59.5</td>
</tr>
<tr>
<td>ρ = 4</td>
<td>64.4 62.8 61.3 60.6 60.5</td>
<td>65.6 64.0 58.3 50.5 47.6</td>
<td>63.8 58.3 53.2 47.2</td>
<td>65.9 65.7 64.5 61.2 59.3</td>
</tr>
<tr>
<td>ρ = 5</td>
<td>64.1 62.5 61.0 60.3 60.3</td>
<td>65.4 63.6 57.6 50.3 47.3</td>
<td>63.6 58.1 52.9 46.9</td>
<td>65.7 65.5 64.3 61.0 59.1</td>
</tr>
</tbody>
</table>

Table 4.7. The mIoUs (in percentage) of segmenting degraded images using different ρ. Let “EWC” denote the segmentation network that employs the EWC module. The five degradation degrees are denoted using d1, d2, d3, d4, and d5, respectively. The numbers with better and the best performance are highlighted in blue and red, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Gaussian Blur</th>
<th>Motion Blur</th>
<th>S &amp; P Noise</th>
<th>Haze</th>
</tr>
</thead>
<tbody>
<tr>
<td>ρ = 1</td>
<td>65.6 63.9 62.2 60.5 60.4</td>
<td>66.4 65.1 59.2 51.3 48.4</td>
<td>64.9 59.8 54.5 48.9</td>
<td>66.7 66.1 64.9 62.1 59.7</td>
</tr>
<tr>
<td>ρ = 2</td>
<td>65.3 63.6 62.0 60.3 60.2</td>
<td>66.3 64.9 59.1 51.1 48.2</td>
<td>64.7 59.5 54.0 48.3</td>
<td>66.6 66.0 64.8 61.8 59.5</td>
</tr>
<tr>
<td>ρ = 3</td>
<td>65.0 63.3 61.7 60.0 59.9</td>
<td>66.0 64.6 59.0 50.9 48.0</td>
<td>64.4 58.9 53.9 47.9</td>
<td>66.4 66.0 64.8 61.6 59.3</td>
</tr>
<tr>
<td>ρ = 4</td>
<td>64.7 63.0 61.5 60.8 60.7</td>
<td>65.8 64.4 58.9 51.0 48.1</td>
<td>64.1 58.6 53.6 47.6</td>
<td>66.1 65.9 64.7 61.4 59.1</td>
</tr>
<tr>
<td>ρ = 5</td>
<td>64.4 62.8 61.3 60.6 60.5</td>
<td>65.6 64.0 58.3 50.5 47.6</td>
<td>63.8 58.1 52.9 46.9</td>
<td>65.9 65.7 64.5 61.0 59.3</td>
</tr>
</tbody>
</table>

using restored images outperforms the proposed DGN. This is because that median filter is very effective in removing salt & pepper noise. Qualitatively, the restored images and the original images are visually identical. Quantitatively, in comparison to the Structure Similarity (SSIM) [113] of the other degradation effects (averagely SSIM = 0.591), the SSIM between the restored image using the median filter and the clean images is 0.863, where SSIM = 1 indicates that two images are completely alike.
identical. Therefore, we can expect the differences between the restored images and the clean image is small, and the segmentation performance of the restored images is high.

Fine-tuning the network using the restored image improves the performance by a large margin. However, the remaining differences between the restored images and the clean images pose an obstacle in improving the performance. To justify this point of view, we train the proposed DGN using the restored images. If there exists no or little gap in the distributions of the features learned using the restored images and the clean images, we can expect the risks of the source and target network to be very similar to each other, such that $\delta_{\text{Gram}} \approx 0$. This indicates that, in comparison to the results fine-tuned using the restored images, we are expecting little or no improvement when training the proposed DGN using the restored images. However, as shown in Table 4.3, the proposed DGN trained on the restored images constantly achieves the best performance and outperforms the approaches fine-tuned on the restored images by averagely 1.9%, 1.0%, 1.9%, and 1.2% for the PASCAL VOC 2012, SUNRGBD, CamVid, and CityScapes datasets, respectively. Therefore, we conclude that the differences between the clean and restored images still hinder the performance from further improvement. The proposed DGN is demonstrated to be effective in degraded image semantic segmentation and can further improve the degraded image semantic segmentation performance when using the restored images.

4.3.4 IMPACT OF GRAM MATRIX & DENSE-INTERWEAVING MATCHING

To validate the impact of the Gram matrix, we conduct the experiments in two respects. Firstly, we train the network (DGN-MSE) by directly minimizing the Mean Square Error (MSE) between the source and target feature maps, without using the Gram matrices, in the dense-interweaving manner. As shown in Table 4.4, when the degradation degree is $d_5$, in comparison to DGN-MSE, the proposed DGN that uses
the Gram matrix improves the segmentation performance by averagely 2.7%.

Secondly, as discussed in [67], matching the Gram matrices can be considered as a maximum mean discrepancy process [7] with the second order polynomial kernel. Similar to [67], we conduct the experiments by adopting 1) linear kernel (DGN-Linear) and 2) Gaussian kernel (DGN-Gaussian) for evaluation. Note that, as the Gram matrix based maximum mean discrepancy and MSE are different in both definition and calculation, the DGN-MSE and DGN-Linear are also different. Quantitatively, as shown in Table 4.4, we achieve comparable segmentation performance when using the different kernels. We conclude that, in the proposed DGN, using either the polynomial (default), the linear, or the Gaussian kernels does not lead to significant changes in segmentation performance.

To validate the impact of the dense-interweaving matching, we modify the proposed DGN and train the network by layer-wisely matching the Gram matrices (DGN-Layerwise), i.e., the Gram matrix of the feature maps of one layer in the target network is matched to its corresponding Gram matrix of the same layer in the source network. In comparison to DGN-Layerwise, the proposed DGN which involves the dense-interweaving matching improves the segmentation performance by averagely 0.6%.

4.3.5 Impact of Hyperparameter $\lambda$ Selection

We conduct experiments to evaluate the impact of hyper-parameter $\lambda$ by using different values $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 10\}$ in Eq. (4.5). As shown in Table 4.5, by increasing $\lambda$, the performance first increases. However, further increasing $\lambda$ would force the network to put more efforts on minimizing the dense-Gram loss, which results in decreasing of the power of optimizing the target network in semantic segmentation task. Based on the results shown in Table 4.5, we select $\lambda = 10^{-1}$ as default. For the other baseline networks, we use the same $\lambda = 10^{-1}$ for all the experiments.
4.3.6 Impact of Dense-Gram Block Selection

We conduct additional experiments to evaluate the segmentation performance when the dense-Gram matching starts at the different convolutional blocks. As the dense-Gram matching tends to force the feature distribution in the target network to be similar to the feature distribution in the source network, the dense-Gram matching can be considered as a form of regularization. Specifically, when the dense-Gram matching starts at a higher block, the proposed DGN allows the target network to learn the features with more freedom, and vice versa.

The quantitative results are shown in Table 4.6. Let “DGN-B2”, “DGN-B3”, “DGN-B4” (default), “DGN-B5”, “DGN-B6” denote the proposed DGN with the dense-Gram matching starting at the 2nd, 3rd, 4th, 5th, and 6th convolutional blocks, respectively. When the degradation degree is small, the dense-Gram matching that starts at a lower block (e.g., DGN-B2) strengthens the ability of the feature regulation and improves the segmentation performance. On the other hand, when the degradation degree is high, the dense-Gram matching that starts at a higher block (e.g., DGN-B6) decreases the ability of feature regulation and decreases the segmentation performance. As shown in Table 4.6, we observe that the DGN-B4 (middle block) provides the best overall performance. For the other segmentation network, we start the dense-Gram matching at the block located in the middle of the network.

4.3.7 Impact of Learning Speed Tuning

Learning speed tuning can be considered as an alternative way of addressing the minimization of the gap in feature distributions of higher layers [57, 74]. Intuitively, to preserve the feature distribution in higher layers, one “naive” way is to manually tune down the learning rate of the higher layers, such that the weight updating speed in higher layers is slow. However, manually tuning the learning rate is heuristic and laborious. A “smart” way of tuning the learning speed is to selectitivity slow down
the learning of network weights by using the Elastic Weight Consolidation (EWC) module [57], such that the network can remember the features learned using the clean images. However, the employment of the EWC module requires additional approximately three times as many parameters as the original network. This level of GPU memory consumption poses a potential obstacle in implementing the EWC modules.

Firstly, we conduct experiments to evaluate the impact of using the smaller learning rates during fine-tuning. For fair comparisons, we only tune down the learning rate of the layers that is associated with the dense-Gram matchings. The learning rate of those higher layers is reduced by multiplying a constant ratio. In this research, we select the ratio to be $\rho \in \{1, 10^{-1}, 10^{-2}, 0\}$, where $\rho = 0$ denotes the weights of higher layers are fixed during fine-tuning and $\rho = 1$ denotes the network using the same learning rate for the whole network fine-tuning. Secondly, we conduct an experiment by employing the EWC module into the segmentation network. We follow the same experiment setting used in network fine-tuning, and select the weight of the EWC loss $\lambda_{\text{EWC}} = 400$ [57].

Quantitatively, as shown in Table 4.7, in comparison to the default network fine-tuning ($\rho = 1$), tuning down the learning rate of the higher blocks with $\rho = 10^{-1}$ improves the performance. However, further tuning down the learning rate (e.g., $\rho = 10^{-2}$ and $\rho = 0$) decreases the performance. In comparison to the fine-tuning based approaches, the employment of EWC module achieves the second best performance. All in all, when the degradation degree is $d_5$, the proposed DGN outperforms the learn rate tuning based approaches by averagely 3.0%, and constantly outperforms the EWC based approach by averagely 1.7%. 
Table 4.8. The mIoUs (in percentage) of segmenting real haze images. The five degradation degrees are denoted using $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>Method</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN8s+fine-tune</td>
<td>56.7</td>
<td>58.3</td>
<td>58.6</td>
<td>56.2</td>
<td>52.7</td>
</tr>
<tr>
<td>FCN8s+C&amp;D</td>
<td>57.3</td>
<td>59.6</td>
<td>57.4</td>
<td>55.3</td>
<td>56.4</td>
</tr>
<tr>
<td>FCN8s+DGN</td>
<td>57.5</td>
<td>58.7</td>
<td>60.2</td>
<td>59.2</td>
<td>56.4</td>
</tr>
<tr>
<td>FCN8s+DGN+C&amp;D</td>
<td>56.9</td>
<td>58.6</td>
<td>56.8</td>
<td>55.1</td>
<td>52.3</td>
</tr>
<tr>
<td>DeepLab v2+fine-tune</td>
<td>60.3</td>
<td>62.4</td>
<td>60.8</td>
<td>62.2</td>
<td></td>
</tr>
<tr>
<td>DeepLab v2+C&amp;D</td>
<td>60.6</td>
<td>60.5</td>
<td>66.8</td>
<td>62.0</td>
<td></td>
</tr>
<tr>
<td>DeepLab v2+DGN</td>
<td>69.2</td>
<td>69.6</td>
<td>69.2</td>
<td>68.2</td>
<td>65.5</td>
</tr>
<tr>
<td>DeepLab v2+DGN+C&amp;D</td>
<td>69.5</td>
<td>69.4</td>
<td>69.7</td>
<td>66.1</td>
<td></td>
</tr>
<tr>
<td>RefineNet+fine-tune</td>
<td>51.1</td>
<td>51.8</td>
<td>50.2</td>
<td>48.5</td>
<td>65.6</td>
</tr>
<tr>
<td>RefineNet+C&amp;D</td>
<td>71.9</td>
<td>71.4</td>
<td>71.2</td>
<td>68.7</td>
<td></td>
</tr>
<tr>
<td>RefineNet+DGN</td>
<td>71.9</td>
<td>71.4</td>
<td>71.4</td>
<td>68.7</td>
<td></td>
</tr>
<tr>
<td>RefineNet+DGN+C&amp;D</td>
<td>72.6</td>
<td>73.2</td>
<td>73.2</td>
<td>72.6</td>
<td>69.4</td>
</tr>
<tr>
<td>EncNet+fine-tune</td>
<td>71.5</td>
<td>71.5</td>
<td>71.4</td>
<td>69.6</td>
<td></td>
</tr>
<tr>
<td>EncNet+C&amp;D</td>
<td>72.6</td>
<td>71.9</td>
<td>71.1</td>
<td>67.0</td>
<td></td>
</tr>
<tr>
<td>EncNet+DGN</td>
<td>72.8</td>
<td>74.0</td>
<td>72.6</td>
<td>71.6</td>
<td></td>
</tr>
<tr>
<td>EncNet+DGN+C&amp;D</td>
<td>72.9</td>
<td>74.0</td>
<td>75.0</td>
<td>74.4</td>
<td>71.2</td>
</tr>
<tr>
<td>DeepLab v3+Fine-tune</td>
<td>76.8</td>
<td>74.5</td>
<td>76.9</td>
<td>74.8</td>
<td>71.5</td>
</tr>
<tr>
<td>DeepLab v3+C&amp;D</td>
<td>79.3</td>
<td>78.7</td>
<td>79.1</td>
<td>76.1</td>
<td>71.5</td>
</tr>
<tr>
<td>DeepLab v3+DGN</td>
<td>78.7</td>
<td>78.9</td>
<td>79.4</td>
<td>77.6</td>
<td>71.1</td>
</tr>
<tr>
<td>DeepLab v3+DGN+C&amp;D</td>
<td>78.8</td>
<td>79.0</td>
<td>78.4</td>
<td>79.7</td>
<td>75.5</td>
</tr>
</tbody>
</table>

4.3.8 Evaluation on Real Haze Images

To further demonstrate the effectiveness of the proposed DGN, we evaluate the proposed method using the 100 real haze images \(^6\) mined from the Internet. Specifically, the mined real haze images are annotated following PASCAL VOC 2012 dataset criteria. For fair comparison, we do not apply the image-restoration processing to the real haze image dataset during the evaluation. Since it is difficult to quantify the degradation degree on the real haze images, we directly deploy five models, denoted by $d_1$, $d_2$, $d_3$, $d_4$, and $d_5$, respectively, that were trained using the corresponding-degree hazy images synthesized from PASCAL VOC 2012 and report their testing performances on the 100 real images in Table 4.8. We note that when degradation degree is $d_3$, the proposed DGN constantly shows the best segmentation performance. We assume that the degradation degree of the real images are similar to the synthesized degree-$d_3$ haze images. As the real haze images are different from the synthesized haze images, we are expecting minor performance decreases.

\(^6\)https://cvl.cse.sc.edu/Download/data_annotated_voc.tar.gz
4.4 Conclusion

In this research, we systematically study the problem of degraded image semantic segmentation and propose a Dense-Gram network to segment degraded images without using any image restoration based pre-processing when only the degraded images are available. The proposed DGN is evaluated using synthetic degraded images based on PASCAL VOC 2012, SUNRGBD, CamVid, and CityScapes benchmark datasets. In comparison to the network fine-tuning based, C&D based, image restoration based, and learning rate tuning based strategies, the proposed DGN substantially improves the semantic segmentation performance of the degraded images.
Chapter 5

Global Context Embedding with Inter-class Shared Boundary Based Encoder

5.1 Motivation

The recent success of deep convolutional neural networks (CNNs) [59, 101, 46, 16, 50, 15, 66] has made remarkable progress in pixel-level semantic segmentation tasks [73, 1, 84, 14, 126]. But the segmentation of small objects is usually inaccurate [56], as small objects usually contribute less to the segmentation loss. For example, as shown in Fig. 5.1(c), sign symbols and poles only take a small fraction of the overall urban street scene, and they could be overlooked in the segmentation. However, accurately segmenting small objects is of great importance in many applications, such as autonomous driving, where driving safety and precise navigation are dependent on the segmentation and recognition of small-sized poles and traffic signs [117, 77, 64, 85, 87, 41]. Here, we take urban street scene segmentation as a study case and develop new CNN-based semantic segmentation methods that can better handle small-sized classes.

One common strategy towards improving the segmentation accuracy of small objects is to increase the scale of input images, to enhance the resolution of small objects, or to produce high-resolution feature maps [63, 72, 117, 64, 77]. This strategy is usually implemented in CNNs by reducing object size induced biases, and training the network to generate multi-scale representation which enhances high-level small-scale features with multiple low-level feature layers. However, those approaches require
data augmentation or increase of the feature dimension. Simply increasing the scale of input images often results in heavy time consumption for both training and testing [117]. The multi-scale representation constructed by the low-level features works as a black-box and cannot guarantee the constructed features are interpretable [64]. Post-processing is another strategy towards improving the accuracy of small object segmentation [58, 14]. As post-processing is not integrated into the segmentation network, the network cannot update its weights according to the post processed results in the training phase [127].

In this research, we propose an Inter-class Shared Boundary based Metric (ISBMetric) to quantify the level of adjacency between each pair of object classes. Specifically, ISBMetric calculates the proportional length of the shared boundaries. For example, ISBMetric between object classes A and B is the proportion between the length of their shared boundaries and the perimeter of A or B. In addition, by quantifying this ratio based level of spatial adjacency, the proposed ISBMetric is robust against object size induced biases, such that small objects can contribute more to the segmentation loss. We demonstrate that the enforcement of the ISBMetric can
help improve the segmentation accuracy of small objects. We propose an ISBMetric based encoder (ISBEncoder) for the purpose. In particular, the proposed ISBEncoder takes the prediction from the segmentation networks as the input and its output is guided by the ISBMetric matrix calculated using segmentation ground truth. In deployment, we only use the trained segmentation network, without the ISBEncoder, to segment new unseen images, such that no extra time or cost is added to the segmentation network. The proposed pipeline – segmentation network with ISBEncoder – is illustrated in Fig. 5.2 and it can be trained in an end-to-end fashion.

We evaluate the proposed method using two urban streetscenes datasets: CamVid [8] and CityScapes [18], and achieve improved results, especially for the small object classes. The effectiveness of the proposed ISBMetric and ISBEncoder is tested and evaluated by combining to many state-of-the-art segmentation networks. To sum up, the main contributions of this research are: 1) We propose a new ISBMetric to measure the level of spatial adjacency between each pair of object classes. The ISBMetric is robust against the object size induced biases, such that small object classes can contribute more to the overall loss. 2) We propose a new ISBEncoder to enforce the ISBMetric in the segmentation of urban streetscenes. The proposed ISBEncoder can be easily combined to many state-of-the-art segmentation networks. 3) We achieve
substantially improved segmentation accuracy of small object classes and improved segmentation accuracy of large object classes using the proposed method without adding extra time or cost during the deployment.

5.2 Method

5.2.1 Overview

The proposed pipeline, as shown in Fig. 5.2, consists of two components: the default segmentation network and the proposed ISBEncoder. Specifically, the default segmentation network can be any network used for semantic segmentation. The proposed ISBEncoder takes the prediction from the segmentation network as input, and its output is guided by the ISBMetric calculated using segmentation ground truth. The overall pipeline can be trained in an end-to-end fashion.

5.2.2 Inter-class Shared Boundary Metric (ISBMetric)

To evaluate the level of spatial adjacency between each pair of object classes, we define the ISBMetric \( m_{i\text{sb}} \) as a \( n_c \times n_c \) matrix with \( n_c \) being the number of object classes, i.e., segmentation labels. This metric is computed from the segmentation map \( s \), where \( s(x, y) \in \{1, 2, \cdots, n_c\} \) is the segmentation-class label at pixel \((x, y)\).

The value of \( m_{i\text{sb}}(i, j) \) is the ratio of the length of the boundary shared by the \( i^{th} \) and the \( j^{th} \) object classes to the \( i^{th} \) object class’ perimeter. Let \( l_i \) denote the \( i^{th} \) object class’ perimeter, and let \( l_{ij} \) denote the length of the shared boundary between the \( i^{th} \) and \( j^{th} \) object classes. The \((i, j)\)-th element in the ISBMetric is \( m_{i\text{sb}}(i, j) = \frac{l_{ij}}{l_i} \), while the value of \((j, i)\)-th element in the ISBMetric is \( m_{i\text{sb}}(j, i) = \frac{l_{ij}}{l_j} \). The value of \( m_{i\text{sb}}(i, i) \) is set to 0 for \( i = 1, 2, \cdots, n_c \). As the perimeters of different object classes are usually different, i.e., \( l_i \neq l_j \) if \( i \neq j \), the ISBMetric \( m_{i\text{sb}} \) is usually asymmetric.

As shown by an example in Fig. 5.3(a), a segmentation map consists of four object classes: 1, 2, 3, 4 with rectangular outer boundary of dimensions 100 × 100, 25 × 25,
Figure 5.3. An illustration of proposed ISBMetric of (a) ground-truth segmentation of an image, and (b-c) two sample segmentation results of the image. Below each segmentation are the corresponding ISBMetric matrix, ISBMetric accuracy and segmentation accuracy.

$25 \times 100$, and $175 \times 175$ pixels, respectively. The ISBMetric $m_{isb}$ is calculated as:

- The perimeter of segmented object 1 is $100 \times 4 + 25 \times 4 = 500$ (combined outer and inner boundaries). The lengths of the boundaries shared by objects 1 and 2, shared by objects 1 and 3, and shared by objects 1 and 4 are 100, 100 and 300, respectively. The first row of the $m_{isb}$ is $[0, \frac{1}{5}, \frac{1}{5}, \frac{3}{5}]$.

- The perimeter of segmented object 2 is 100, and object 2 is fully enclosed by object 1 and is not adjacent to objects 3 and 4. The length of the boundaries shared by objects 1 and 2 is 100. The second row of the $m_{isb}$ is $[1, 0, 0, 0]$.

- The perimeter of segmented object 3 is 250. The object 3 is not adjacent to object 2. The lengths of boundaries shared by objects 1 and 3, and shared by object 3 and 4 are 100 and 150. The third row of the $m_{isb}$ is $[\frac{2}{5}, 0, 0, \frac{3}{5}]$.

- The perimeter of segmented object 4 is 1,150. The object 4 is not adjacent to object 2. The lengths of boundaries shared by objects 1 and 4, and shared by objects 3 and 4 are 300 and 150. The fourth row of the $m_{isb}$ is $[\frac{6}{23}, 0, \frac{3}{23}, 0]$.

When both the spatial adjacency between the object classes and object size are changed, as illustrated in Fig. 5.3(b) – right-shift the object 3 by 1 pixel and enlarge
the object 2 to a size of 50 × 50, ISBMetric $m_{isb}$ will be changed as follow:

- The perimeter of segmented object 1 is changed to 600. The objects 1 and 3 are no longer adjacent, and the length of the boundaries shared by them is changed to 0. The first row of the $m_{isb}$ is changed to $[0, \frac{1}{3}, 0, \frac{2}{3}]$.

- The object 2 is still enclosed by object 1. Although the object 2 is enlarged to a size of 50 × 50, the second row of the $m_{isb}$ is still $[1, 0, 0, 0]$.

- The object 3 is not adjacent to objects 1 and 2, and is fully enclosed by object 4. The third row of the $m_{isb}$ is changed to $[0, 0, 0, 1]$.

- The perimeter of segmented object 4 is changed to 1,350. The fourth row of the $m_{isb}$ is changed to $[\frac{8}{27}, 0, \frac{5}{27}, 0]$.

We can see that the proposed ISBMetric is highly sensitive to the spatial adjacency changes of objects but is very robust against the object size induced biases, e.g., the value of this metric does not rely much on the object size. This way, the small object classes can contribute more to the segmentation loss, which helps improve the segmentation accuracy of the small object classes. We adopt the 4-connectivity neighboring system to measure the boundaries of object classes, where the 4-connectivity neighboring system [33] is defined regarding pixel neighborhoods. For a pixel at $(x, y)$, a 4-connectivity neighboring $\{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$ contains only
the pixels above, below, to the left and to the right of the center pixel \((x, y)\). The boundary length is calculated considering the boundary pixel’s 4 neighbors.

### 5.2.3 Rationale of the ISBMetric Based Encoder (ISBEncoder)

The ISBMetric is calculated based on the prediction from the segmentation network. For a \(3 \times h \times w\) input RGB image, the prediction of the segmentation network is a \(n_c \times h \times w\) matrix. Each pixel in the prediction map contains a set of probabilities of this pixel being in class \(c \in \{1, \cdots, n_c\}\).

One obstacle towards implementing the proposed ISBMetric is that, in the training phase, it first needs to convert the predicted set of probabilities to a discrete class label, such that each pixel only has one class label. This way, the boundaries of the classes can be determined.

Let \(s_{\text{pred}}^{\text{c}}(c, x, y)\) denote the probability of the \(c^{\text{th}}\) class at pixel \((x, y)\). The discrete-class label of pixel \((x, y)\) is determined by the index of the maximum value of the predicted class probabilities:

\[
c^* = \arg\max_{c \in \{1, \cdots, n_c\}} s_{\text{pred}}^{\text{c}}(c, x, y),
\]

where \(c^*\) denotes the index of the maximum value.

Using gradient descent based optimization approach for network parameters updating, the partial derivative of the forward propagation function w.r.t. the network parameters must exist [90]. However, the derivative of the Eq. (5.1) w.r.t. the index \(c \in \{1, \cdots, n_c\}\) does not exist [106]. Therefore, the partial derivatives of the ISBMetric loss w.r.t. the associated network parameters cannot be retrieved. As a result, we **cannot** directly employ the ISBMetric to the network.

To circumvent this dilemma, we train a separate network to simulate the calculation of the proposed ISBMetric by taking \(s_{\text{pred}}^{\text{c}}\) as the input. The ISBEncoder works as an add-on component to the segmentation network, which aims to calculate the ISBMetric \(m_{\text{isb}}\) using the predictions (before the loss layer) from the segmentation...
Table 5.1. Detailed configuration of the proposed ISBEncoder architecture. We use the image size of $360 \times 480$ for demonstration.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Kernel Size</th>
<th>Stride</th>
<th>Pad</th>
<th>Output Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$n_c \times 360 \times 480$</td>
</tr>
<tr>
<td>conv1_1</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>64 x 360 x 480</td>
</tr>
<tr>
<td>conv1_2</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>64 x 360 x 480</td>
</tr>
<tr>
<td>pool1</td>
<td>2 x 2</td>
<td>2</td>
<td>0</td>
<td>64 x 180 x 240</td>
</tr>
<tr>
<td>conv2_1</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>128 x 180 x 240</td>
</tr>
<tr>
<td>conv2_2</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>128 x 180 x 240</td>
</tr>
<tr>
<td>pool2</td>
<td>2 x 2</td>
<td>2</td>
<td>0</td>
<td>128 x 90 x 120</td>
</tr>
<tr>
<td>conv3_1</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>256 x 90 x 120</td>
</tr>
<tr>
<td>conv3_2</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>256 x 90 x 120</td>
</tr>
<tr>
<td>conv3_3</td>
<td>3 x 3</td>
<td>1</td>
<td>1</td>
<td>256 x 90 x 120</td>
</tr>
<tr>
<td>pool3</td>
<td>2 x 2</td>
<td>2</td>
<td>0</td>
<td>256 x 45 x 60</td>
</tr>
<tr>
<td>conv4</td>
<td>1 x 1</td>
<td>1</td>
<td>0</td>
<td>32 x 45 x 60</td>
</tr>
<tr>
<td>fc5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4096</td>
</tr>
<tr>
<td>fc6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4096</td>
</tr>
<tr>
<td>fc7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$n_c \times n_c$</td>
</tr>
</tbody>
</table>

network, i.e., the ISBEncoder extends the original segmentation network by adding a sub-network to perform a new task – ISBMetric estimation. The loss between $m_{\text{pred}}^{\text{isb}}$ and $m_{\text{gt}}^{\text{isb}}$ affects the parameter tuning in the segmentation network. As the loss between the $m_{\text{pred}}^{\text{isb}}$ and $m_{\text{gt}}^{\text{isb}}$ highlights the small object classes, the segmentation network would be forced to put more emphasis on correctly segmenting the small object classes.

5.2.4 ISBEncoder Architecture

The ISBEncoder network architecture is modified from the VGG-16 [101] network. The ISBEncoder takes the $n_c$-channel prediction from the segmentation network as input. This is followed by a series of three convolution blocks (1, 2 and 3), as denoted in dashed brown boxes in Fig. 5.4, one bottleneck layer [50], and three fully connected layers. The detailed layer-wise settings are reported in Table 5.1.

All convolutional layers in the ISBEncoder are followed by the Rectified Linear Unit (ReLU) non-linear activation layer to introduce element-wise non-linearity [79]. To reduce the feature dimensions, alleviate the memory demand and accelerate the training process [50], we introduce a bottleneck layer using $1 \times 1$ convolution kernel with stride 1 and padding 0 after the third convolutional block (3). Three fully-
connected layers are applied after the bottleneck layer: the first two have 4,096 channels each, the third performs $n_c \times n_c$-way ISBMetric prediction and thus outputs $n_c \times n_c$ channels (one for each element in the ISBMetric $m_{isb}$).

In training, the weights of the kernels in the three convolutional blocks (1, 2 and 3) are initialized from the VGG-16 net. The weights of bottleneck layer and fully-connected layers are initialized using Xavier initialization [36]. The output of the third fully-connected layer is then reshaped to a $n_c \times n_c$ matrix to match the dimension of the ground truth ISBMetric. The last layer is the mean-square error loss layer, which is used to calculate the distances between the predicted ISBMetric and the ground truth ISBMetric. The experimental evaluation of the ISBEncoder will be reported in Sections 5.3.2 and 5.3.2.

5.2.5 The Overall Pipeline

In the phase of pipeline training, the pipeline weights are optimized based on 1) segmentation loss and 2) ISBEncoder loss. For the segmentation network, we follow [73] and adopt the sigmoid cross-entropy loss for training:

$$L_{seg} = -\frac{1}{n_c \cdot h \cdot w} \sum_{c=1}^{n_c} \sum_{y=1}^{h} \sum_{x=1}^{w} \left( s_{gt}^{c}(c, x, y) \log s_{pred}^{c}(c, x, y) + (1 - s_{gt}^{c}(c, x, y)) \log(1 - s_{pred}^{c}(c, x, y)) \right),$$  \hspace{1cm} (5.2)

where $s_{gt}^{c}(c, x, y)$ denote the ground truth (0 or 1) of class $c$ at pixel $(x, y)$. For the ISBEncoder, we use mean square error (MSE) loss for training:

$$L_{isb} = \frac{1}{n_c \cdot n_c} \sum_{i=1}^{n_c} \sum_{j=1}^{n_c} \left( m_{isb}^{pred}(i, j) - m_{isb}^{gt}(i, j) \right)^2, \hspace{1cm} (5.3)$$

where $m_{isb}^{pred}$ and $m_{isb}^{gt}$ are the predicted and ground truth ISBMetric matrices, respectively. Then, the overall loss combines segmentation network loss and the ISBEncoder loss:

$$L = L_{seg} + \lambda L_{isb}, \hspace{1cm} (5.4)$$
Figure 5.5. Examples of semantic segmentation results on CamVID (left) and CityScapes (right) validation datasets. For visualization purpose, the ground-truth segmentation is superimposed to the input image, and the dashed rectangles are enlarged for highlighting improvements.

Table 5.2. The comparison results of small object classes (left) and large object classes (right) on CamVid testing dataset (in percentage). For each object class, the numbers with better and the best performance are highlighted in blue and red, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Sign</th>
<th>Pedestrian</th>
<th>Pole</th>
<th>Bicyclist</th>
<th>mIoU</th>
<th>Building</th>
<th>Tree</th>
<th>Sky</th>
<th>Car</th>
<th>Road</th>
<th>Sidewalk</th>
<th>Fence</th>
<th>mIoU_s</th>
<th>mIoU_l</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALE</td>
<td>24.4</td>
<td>29.1</td>
<td>13.6</td>
<td>28.6</td>
<td>21.9</td>
<td>73.4</td>
<td>70.2</td>
<td>91.1</td>
<td>64.2</td>
<td>91.1</td>
<td>72.4</td>
<td>31.0</td>
<td>70.5</td>
<td>53.6</td>
</tr>
<tr>
<td>Liu &amp; He</td>
<td>24.4</td>
<td>28.0</td>
<td>8.3</td>
<td>8.5</td>
<td>16.6</td>
<td>66.3</td>
<td>66.6</td>
<td>98.4</td>
<td>62.3</td>
<td>85.8</td>
<td>64.5</td>
<td>17.5</td>
<td>64.8</td>
<td>47.2</td>
</tr>
<tr>
<td>SuperParsing</td>
<td>25.4</td>
<td>11.6</td>
<td>5.2</td>
<td>8.9</td>
<td>12.8</td>
<td>70.4</td>
<td>54.8</td>
<td>83.5</td>
<td>43.1</td>
<td>83.4</td>
<td>37.4</td>
<td>18.3</td>
<td>58.7</td>
<td>42.0</td>
</tr>
<tr>
<td>SegNet</td>
<td>13.4</td>
<td>25.3</td>
<td>16.0</td>
<td>24.8</td>
<td>19.9</td>
<td>68.7</td>
<td>52.9</td>
<td>87.0</td>
<td>58.5</td>
<td>86.2</td>
<td>60.5</td>
<td>17.9</td>
<td>61.5</td>
<td>46.4</td>
</tr>
<tr>
<td>SegNet + ISBMetric-w</td>
<td>19.2</td>
<td>25.7</td>
<td>18.3</td>
<td>26.4</td>
<td>22.4</td>
<td>71.7</td>
<td>51.8</td>
<td>86.2</td>
<td>58.1</td>
<td>81.9</td>
<td>62.3</td>
<td>20.1</td>
<td>61.7</td>
<td>47.4</td>
</tr>
<tr>
<td>SegNet + ISBEncoder</td>
<td>19.7</td>
<td>26.8</td>
<td>19.2</td>
<td>28.1</td>
<td>23.5</td>
<td>72.4</td>
<td>52.4</td>
<td>86.8</td>
<td>57.3</td>
<td>83.3</td>
<td>63.7</td>
<td>22.7</td>
<td>62.7</td>
<td>48.4</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>32.4</td>
<td>32.0</td>
<td>11.9</td>
<td>37.4</td>
<td>28.4</td>
<td>76.4</td>
<td>71.2</td>
<td>87.9</td>
<td>75.0</td>
<td>75.0</td>
<td>72.2</td>
<td>56.7</td>
<td>72.9</td>
<td>56.7</td>
</tr>
<tr>
<td>FCN-8s + ISBMetric-w</td>
<td>37.2</td>
<td>35.1</td>
<td>15.7</td>
<td>43.1</td>
<td>33.5</td>
<td>75.9</td>
<td>71.1</td>
<td>87.8</td>
<td>78.2</td>
<td>75.1</td>
<td>73.6</td>
<td>36.1</td>
<td>73.6</td>
<td>59.1</td>
</tr>
<tr>
<td>FCN-8s + ISBEncoder</td>
<td>37.6</td>
<td>38.5</td>
<td>16.8</td>
<td>42.5</td>
<td>33.9</td>
<td>81.8</td>
<td>70.1</td>
<td>87.7</td>
<td>78.9</td>
<td>91.4</td>
<td>77.2</td>
<td>37.0</td>
<td>74.9</td>
<td>60.0</td>
</tr>
<tr>
<td>DLA</td>
<td>48.8</td>
<td>38.6</td>
<td>27.8</td>
<td>55.4</td>
<td>47.7</td>
<td>83.2</td>
<td>77.2</td>
<td>91.2</td>
<td>83.6</td>
<td>94.3</td>
<td>81.4</td>
<td>32.0</td>
<td>77.5</td>
<td>66.7</td>
</tr>
<tr>
<td>DLA-34 + ISBMetric-w</td>
<td>50.8</td>
<td>60.1</td>
<td>29.2</td>
<td>55.9</td>
<td>49.0</td>
<td>81.7</td>
<td>75.6</td>
<td>90.9</td>
<td>84.3</td>
<td>93.2</td>
<td>81.8</td>
<td>33.6</td>
<td>77.3</td>
<td>67.0</td>
</tr>
<tr>
<td>DLA-34 + ISBEncoder</td>
<td>51.2</td>
<td>60.3</td>
<td>29.1</td>
<td>56.8</td>
<td>49.4</td>
<td>82.1</td>
<td>75.9</td>
<td>91.1</td>
<td>84.1</td>
<td>93.9</td>
<td>82.3</td>
<td>34.7</td>
<td>77.7</td>
<td>67.4</td>
</tr>
</tbody>
</table>

where $\lambda$ is a balance coefficient. The empirical selection of $\lambda$ is discussed in Section 5.3.2. The proposed pipeline is trained in an end-to-end fashion.
5.3 Experiments

5.3.1 Implementation Details

The proposed method is implemented using Caffe\(^1\) based SegNet\(^2\), PyTorch\(^3\) based FCN\(^4\), GCN\(^5\), DLA\(^6\), and Tensorflow\(^7\) based DeepLab\(^8\) and PSPNet\(^9\) with Intel Core i7 6700K, and with NVIDIA 1080 Ti GPU with mini-batch Stochastic Gradient Descent (SGD).

The overall pipeline consists of two components: 1) Segmentation network, 2) ISBEncoder. The segmentation network weights are initialized from their pre-trained models. The segmentation network and ISBEncoder are jointly trained using SGD with momentum of 0.9, weight decay of 0.0001 and adaptive learning rates. The mini-batch size is 3 when using the CamVid dataset for the network training, and the mini-batch size is 1 when using CityScapes dataset for the network training. We initially set the learning rate as suggested in [73, 1, 84, 126, 15]. We use the “poly” learning rate policy [14], where the initial learning rate is multiplied by \(\left(1 - \frac{\text{iter}}{\text{max\_iter}}\right)^{\text{power}}\) with \(\text{power} = 0.9\) [126]. The number of overall training iterations is 20k for both CamVid and CityScapes datasets.

\(^1\)http://caffe.berkeleyvision.org/
\(^2\)https://github.com/alexgkendall/caffe-segnet
\(^3\)https://github.com/pytorch
\(^4\)https://github.com/wkentaro/pytorch-fcn
\(^5\)https://github.com/zijundeng/pytorch-semantic-segmentation
\(^6\)https://github.com/ucbdrive/dla
\(^7\)https://www.tensorflow.org/
\(^8\)https://github.com/tensorflow/models/tree/master/research/deeplab
\(^9\)https://github.com/holyseven/PSPNet-TF-Reproduce
Table 5.2. The comparison results of small object classes (top) and large object classes (bottom) on CityScapes testing dataset (in percentage). For each object class, the numbers with better and the best performance are highlighted in blue and red, respectively.

Comparisons to Baselines and Existing Methods

To demonstrate the effectiveness of the proposed method, we evaluate the proposed method using the baseline segmentation networks with spatial pyramid pooling-based architecture (e.g., FCN-8s [73], SegNet [1], GCN [84], PSPNet [126], and DeepLabV3 [15]), and the baseline segmentation network with feature pyramid network-based architecture (e.g., DLA [?]). For CamVid dataset, three baseline segmentation networks – FCN-8s, SegNet, and DLA – are trained and evaluated. For CityScapes
dataset, five baseline segmentation networks – FCN-8s, GCN, PSPNet, DeepLabV3, and DLA are trained and evaluated. The proposed pipeline is also compared with several existing segmentation methods: ALE [94], SuperParsing [109], Liu & He [71], Deeplab-LFOV [14], and FoveaNet [66].

Using CamVid dataset for evaluation, the quantitative results are shown in Table 5.2. We observe that by employing the proposed ISBEncoder to the baseline segmentation network, the IoU scores of the small objects classes can be significantly improved when comparing to the settings without the ISBEncoder. Combining the ISBEncoder to the SegNet, FCN-8s, and DLA baseline segmentation networks, it shows 3.6%, 5.5% and 1.7% improvements for small-object classes (mIoU<sub>S</sub>), respectively. It also shows 1.2%, 2.0% and 0.2% improvements for large-object classes (mIoU<sub>L</sub>) using the SegNet, FCN-8s, and DLA respectively. The overall mIoU improvements are 2.0% for SegNet, 3.3% for FCN-8s, and 0.7% for DLA.

Using CityScapes dataset for evaluation, the quantitative results are shown in Table 5.2. We also observe significant improvements on segmenting small object classes after employing the ISBEncoder. It shows 3.2%, 3.0%, 2.8%, 1.7%, and 2.2% improvements for small-object classes (mIoU<sub>S</sub>) using FCN-8s, GCN, PSPNet, DeepLab V3, and DLA, respectively. It shows 1.6%, 0.3%, 0.3%, 0.03%, and 0.04% improvements for large-object classes (mIoU<sub>L</sub>) using FCN-8s, GCN, PSPNet, DeepLab V3, and DLA, respectively. The overall mIoU improvements by including ISBEncoder are 2.2% for FCN-8s, 1.3% for GCN, 1.3% for PSPNet, 0.7% for DeepLab V3, and 0.8% for DLA.

To demonstrate the effectiveness of the proposed ISBMetric, we conduct an additional experiment using the weighted loss function whose weights are based on the ISBMetric. The conventional weighted loss function used in semantic segmentation, e.g., weighted sigmoid cross-entropy loss, requires a single value as the weight for each object class. However, the weight for each object class in ISBMetric is a row
vector, which makes it difficult to be directly applied to the weighted loss function. Alternatively, as each row in the ISBMetric is associated with an object class, we calculate the row-wise MSE between the ISBMetric of the segmentation prediction and the ISBMetric ground-truth, and use the calculated row-wise Mean Square Error (row-wise MSE) to weigh each object class in the segmentation loss function (weighted sigmoid cross-entropy loss). In the experiment, we first calculate the ISBMetric using the segmentation predictions. Then, we calculate the row-wise MSE based on $m_{isb}^{pred}$ and $m_{isb}^{gt}$, and use the calculated row-wise MSE as the weight of the object class to train the segmentation network. Experimental results are shown in Tables 5.2 and 5.2, in which “ISBMetric-w” denotes the method uses the ISBMetric based weighted loss function. Using CamVid testing dataset for evaluation, the experimental results demonstrate that weighing the object classes using the row-wise MSE of the ISBMetric shows 2.5%, 5.1%, and 1.3% mIoU $S$ improvements, and 1.0%, 2.3%, and 0.4% mIoU improvements for the SegNet, FCN-8s, and DLA, respectively. Using CityScapes testing dataset for evaluation, it shows shows 2.5%, 2.2%, 2.2%, 1.3%, and 1.6% mIoU $S$ improvements, and 1.6%, 0.8%, 0.9%, 0.3%, and 0.4% mIoU improvements for the FCN-8s, GCN, PSPNet, DeepLab v3, and DLA, respectively. In comparison to the proposed method, the segmentation performance of small object classes when using the ISBMetric based weighted loss function is better than the baselines but is slightly worse than the proposed method.

To visually demonstrate the effectiveness of the proposed ISBEncoder, we provide representative segmentation results of FCN-8s and PSPNet with or without ISBEncoder on CamVid and CityScapes datasets in Fig. 5.5. For small-object classes, we find that the regions segmented using ISBEncoder are more accurate, e.g., the poles, sign symbols and person, indicated by dashed rectangles, which are insufficiently segmented or totally missing when using the baseline method. By employing proposed ISBEncoder to the baseline segmentation network, it can better capture the missing
components and render more accurate segmentation results.

**Evaluation of the ISBEncoder Accuracy**

Table 5.3. The ISBEncoder mean square errors of the small object classes (MSE$_S$), the large object classes (MSE$_L$), and all classes (MSE) on CamVid and CityScapes testing datasets.

<table>
<thead>
<tr>
<th>Input</th>
<th>Dataset</th>
<th>MSE$_S$</th>
<th>MSE$_L$</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^gt$</td>
<td>CamVid</td>
<td>6.08 x 10^{-9}</td>
<td>6.09 x 10^{-5}</td>
<td>6.09 x 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>CityScapes</td>
<td>5.88 x 10^{-5}</td>
<td>5.89 x 10^{-5}</td>
<td>5.89 x 10^{-5}</td>
</tr>
<tr>
<td>$s^{disc}$</td>
<td>CamVid</td>
<td>6.12 x 10^{-9}</td>
<td>6.13 x 10^{-5}</td>
<td>6.13 x 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>CityScapes</td>
<td>5.91 x 10^{-5}</td>
<td>5.92 x 10^{-5}</td>
<td>5.92 x 10^{-5}</td>
</tr>
<tr>
<td>$s^{pred}$</td>
<td>CamVid</td>
<td>6.32 x 10^{-9}</td>
<td>6.34 x 10^{-5}</td>
<td>6.33 x 10^{-5}</td>
</tr>
<tr>
<td></td>
<td>CityScapes</td>
<td>5.97 x 10^{-5}</td>
<td>5.99 x 10^{-5}</td>
<td>5.98 x 10^{-5}</td>
</tr>
</tbody>
</table>

Figure 5.6. An illustration of the predicted ISBMetric $m^{pred}_{istb}$ and the ground truth ISBMetric $m^{gt}_{istb}$ of a sample image from CityScapes training dataset. The intensity of the colorbar denotes the value.

We conduct three experiments to evaluate how accurate the ISBEncoder can simulate the ISBMetric matrix calculation: 1) We use the segmentation ground truth

Table 5.4. The comparison results of small object classes (left) and large object classes (right) using the proposed pipeline with different hyperparameter $\lambda$ (in Eq. (5.4)) on CamVid testing dataset (in percentage). The baseline segmentation network is FCN-8s. For each object class, the numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Sign symbol</th>
<th>Pedestrian</th>
<th>Pole</th>
<th>Bicycle</th>
<th>mIoU$_S$</th>
<th>Building</th>
<th>Tree</th>
<th>Sky</th>
<th>Car</th>
<th>Road</th>
<th>Sidewalk</th>
<th>Fence</th>
<th>mIoU$_L$</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32.4</td>
<td>32.3</td>
<td>11.9</td>
<td>37.1</td>
<td>28.4</td>
<td>76.3</td>
<td>71.4</td>
<td>87.9</td>
<td>78.0</td>
<td>91.9</td>
<td>73.0</td>
<td>32.2</td>
<td>72.9</td>
<td>56.7</td>
</tr>
<tr>
<td>1</td>
<td>36.9</td>
<td>37.6</td>
<td>12.6</td>
<td>42.0</td>
<td>32.3</td>
<td>81.3</td>
<td>70.0</td>
<td>87.7</td>
<td>79.2</td>
<td>91.6</td>
<td>77.4</td>
<td>36.7</td>
<td>74.8</td>
<td>59.4</td>
</tr>
<tr>
<td>2</td>
<td>37.6</td>
<td>38.3</td>
<td>16.8</td>
<td>42.5</td>
<td>33.9</td>
<td>81.8</td>
<td>70.1</td>
<td>87.9</td>
<td>79.9</td>
<td>91.4</td>
<td>77.2</td>
<td>37.0</td>
<td>74.9</td>
<td>60.0</td>
</tr>
<tr>
<td>4</td>
<td>38.4</td>
<td>38.4</td>
<td>17.2</td>
<td>42.3</td>
<td>33.9</td>
<td>81.8</td>
<td>68.9</td>
<td>87.3</td>
<td>79.1</td>
<td>91.4</td>
<td>77.3</td>
<td>36.6</td>
<td>74.7</td>
<td>59.8</td>
</tr>
<tr>
<td>8</td>
<td>40.0</td>
<td>38.5</td>
<td>14.4</td>
<td>40.2</td>
<td>32.7</td>
<td>80.2</td>
<td>69.8</td>
<td>86.1</td>
<td>77.9</td>
<td>91.4</td>
<td>77.1</td>
<td>35.7</td>
<td>74.0</td>
<td>59.0</td>
</tr>
<tr>
<td>16</td>
<td>39.7</td>
<td>35.3</td>
<td>14.4</td>
<td>40.7</td>
<td>32.9</td>
<td>76.0</td>
<td>69.4</td>
<td>86.1</td>
<td>75.0</td>
<td>91.0</td>
<td>66.6</td>
<td>36.1</td>
<td>73.4</td>
<td>58.3</td>
</tr>
</tbody>
</table>
Table 5.5. The comparison results of small object classes (top) and large object classes (bottom) using proposed pipeline with different hyper-parameter $\lambda$ (in Eq. (5.4)) on CityScapes testing dataset (in percentage). The baseline segmentation network is PSPNet. For each object class, the numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>pole</th>
<th>traffic light</th>
<th>traffic sign</th>
<th>person</th>
<th>rider</th>
<th>motorcycle</th>
<th>bicycle</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$= 0$</td>
<td>62.9</td>
<td>69.7</td>
<td>77.7</td>
<td>80.8</td>
<td>61.8</td>
<td>66.0</td>
<td>72.8</td>
<td>71.0</td>
</tr>
<tr>
<td>$= 1$</td>
<td>64.8</td>
<td>67.7</td>
<td>79.2</td>
<td>82.1</td>
<td>63.6</td>
<td>66.7</td>
<td>79.4</td>
<td>72.4</td>
</tr>
<tr>
<td>$= 0$</td>
<td>65.0</td>
<td>71.2</td>
<td>81.1</td>
<td>82.4</td>
<td>64.4</td>
<td>66.9</td>
<td>79.6</td>
<td>72.9</td>
</tr>
<tr>
<td>$= 4$</td>
<td>66.9</td>
<td>72.6</td>
<td>82.2</td>
<td>82.9</td>
<td>64.7</td>
<td>67.5</td>
<td>79.8</td>
<td>75.7</td>
</tr>
<tr>
<td>$= 8$</td>
<td>66.8</td>
<td>72.8</td>
<td>82.4</td>
<td>83.2</td>
<td>64.5</td>
<td>67.7</td>
<td>79.9</td>
<td>73.8</td>
</tr>
<tr>
<td>$= 16$</td>
<td>65.6</td>
<td>69.9</td>
<td>80.9</td>
<td>81.8</td>
<td>63.9</td>
<td>66.5</td>
<td>79.8</td>
<td>72.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>road</th>
<th>sidewalk</th>
<th>building</th>
<th>wall</th>
<th>fence</th>
<th>vegetation</th>
<th>tree</th>
<th>sky</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$= 0$</td>
<td>98.2</td>
<td>86.4</td>
<td>92.9</td>
<td>58.4</td>
<td>62.4</td>
<td>91.6</td>
<td>61.3</td>
<td>94.3</td>
<td>95.4</td>
</tr>
<tr>
<td>$= 1$</td>
<td>98.1</td>
<td>85.9</td>
<td>91.5</td>
<td>60.2</td>
<td>64.1</td>
<td>90.7</td>
<td>64.9</td>
<td>94.4</td>
<td>94.8</td>
</tr>
<tr>
<td>$= 2$</td>
<td>98.1</td>
<td>85.9</td>
<td>91.5</td>
<td>60.2</td>
<td>64.4</td>
<td>90.7</td>
<td>65.7</td>
<td>94.4</td>
<td>94.8</td>
</tr>
<tr>
<td>$= 4$</td>
<td>98.1</td>
<td>86.4</td>
<td>91.8</td>
<td>56.9</td>
<td>64.9</td>
<td>91.4</td>
<td>66.2</td>
<td>94.6</td>
<td>95.2</td>
</tr>
<tr>
<td>$= 8$</td>
<td>98.1</td>
<td>85.5</td>
<td>91.7</td>
<td>57.4</td>
<td>65.4</td>
<td>91.1</td>
<td>66.3</td>
<td>94.3</td>
<td>95.2</td>
</tr>
<tr>
<td>$= 16$</td>
<td>98.0</td>
<td>86.3</td>
<td>91.6</td>
<td>56.7</td>
<td>64.7</td>
<td>90.8</td>
<td>65.6</td>
<td>93.5</td>
<td>94.9</td>
</tr>
</tbody>
</table>

$s^{gt}$ as the input. The output $m_{ib}^{pred}$ is then compared to the $m_{ib}^{gt}$. 2) We first convert the prediction map $s^{pred}$, which is generated from the segmentation network, to its discrete map $s^{disc}$ using Eq. (5.1). And then, we follow the same procedures described in Section 5.2.2 to calculate the $m_{ib}^{disc}$ using the discrete map. We also use the converted discrete map as the input of the ISBEncoder, and compare the output $m_{ib}^{pred}$ to the ISBMetric $m_{ib}^{disc}$. 3) We directly use the prediction map $s^{pred}$ as the input of the ISBEncoder. The output $m_{ib}^{pred}$ is then compared to the ISBMetric $m_{ib}^{disc}$.

To evaluate the accuracy of the ISBEncoder, we use Eq. (5.3) to calculate the mean square error between the ISBMetric matrix predicted using the ISBEncoder and the ISBMetric matrix calculated using the procedures described in Section 5.2.2. A sample $m_{ib}^{pred}$ and $m_{ib}^{gt}$ are illustrated in Fig. 5.6. As shown in Table 5.3, using 1) the segmentation ground truth maps, 2) the discrete maps, and 3) the segmentation prediction maps as the inputs of the ISBEncoder, the mean square errors of the small object classes ($\text{MSE}_S$), the large object classes ($\text{MSE}_L$), and all classes ($\text{MSE}$) are all very small.

To evaluate the performance of the ISBEncoder in handling the object translation
in the image, we conduct an experiment on a synthetic 2D shapes dataset [116]. Particularly, the synthetic dataset contains three objects of different shapes: Circle, square, and triangles. There are 20,000 paired images for training and 500 for testing. In each paired images, the same shapes are of the same size and color but of different locations and occlusions. In the images of different pairs, the sizes of the same shapes are chosen randomly. We train the whole pipeline using the synthetic dataset. Quantitatively, we calculate the MSE between $m_{isb}^{pred}$ and $m_{isb}^{gt}$ for each images, it yields a mean $\text{MSE}_{2D} = 1.64 \times 10^{-5}$. The experimental results show that the mean $\text{MSE}_{2D}$ is very small. As the predicted $m_{isb}^{pred}$ is consistent with the ground-truth $m_{isb}^{gt}$, we conclude that the proposed ISBEncoder can accurately capture location variations.

**Impact of the ISBEncoder**

To evaluate the impact of the ISBEncoder, we first conduct experiments using different hyper-parameter values $\lambda \in \{0, 1, 2, 4, 8, 16\}$. Secondly, we qualitatively evaluate the feature map with or without the ISBEncoder. Thirdly, we demonstrate the impact of the ISBEncoder during network training.

Firstly, from the results shown in Tables 5.4 and 5.5, for small object classes, we find that, by increasing $\lambda$, the segmentation accuracy first increases to a maximum value and then slightly decrease. This can also be observed in Fig. 5.5, where the ability of predicting small objects is first boosted along with increased $\lambda$ by weighting more on the ISBMetric loss. However, further increasing $\lambda$ will cause a possible increase in false positives. For example, in the second column of Fig. 5.5, the poles in the white dashed-rectangle are insufficiently segmented by the baseline segmentation network. By increasing $\lambda$ (from 1 to 4), the network shows an improved performance on capturing more pole pixels. However, keep increasing $\lambda$ (from 8 to 16), the network

---

10http://visualdynamics.csail.mit.edu/
Figure 5.7. An illustration of the impact of the proposed ISBEncoder on the feature map after the “conv6”/the last convolutional layer in the PSPNet. The input image is from the CityScapes training dataset.

Figure 5.8. (a) The mIoU of large object classes, overall classes, and small object classes over 20k training iterations on CityScapes training dataset (in percentage). (b) The mean square error (MSE) loss of ISBEncoder over 20k training iterations. The baseline segmentation network is PSPNet.

mistakenly classify the pixels around the poles in the building as poles, such that the poles become visually thicker as $\lambda$ increases.

Secondly, we demonstrate the impact of the ISBEncoder by visualizing a sample feature map (after “conv6”/the last convolutional layer in the PSPNet) using a sample training image from the CityScapes dataset. As shown in Fig. 5.7(c), most of the small objects are not highlighted in the feature map when using the baseline segmentation network. Whereas, as shown in Fig. 5.7(d), the small objects are better highlighted
in the feature map when employing the ISBEncoder to the segmentation network.

Thirdly, we demonstrate impact of the ISBEncoder during training. The ISBEncoder convergence is shown in Fig. 5.8, from which we can see that mIoU_S and mIoU_L are improved with the increase of iterations. The ISBEncoder MSE loss is converged after 8k iterations. We observe that the improvement of mIoU_S is more significant, while the improvement of mIoU_L is smaller. In summary, both qualitative and quantitative results verify that the proposed ISBEncoder can effectively improve the segmentation accuracy of small objects.

5.4 Conclusion

This research proposed an ISBMetric to measure the level of spatial adjacency between each pair of object classes, and proposed an ISBEncoder to enforce the ISBMetric in the segmentation of urban street scene. Based on the experiment results, the proposed method can substantially improve the segmentation accuracy of small objects, as well as improve the overall segmentation performance. The proposed ISBMetric is evaluated based on FCN-8s, SegNet, GCN, PSPNet, DeepLab V3, and DLA networks on CamVid and CityScapes datasets.
6.1 Motivation

Most of the current state-of-the-art semantic segmentation models are originated from the Fully Convolutional Network (FCN) [73], which conducts image-to-image dense per-pixel predictions for multi-class semantic segmentation. Despite the success of the FCN-based segmentation networks, one major challenge of the existing networks is its difficulty in handling training data that are imbalanced and exhibit highly-skewed class distributions. Ideally, these segmentation networks expect balanced class data or equal misclassification costs [43, 23]. When presented with complex imbalanced training data, these segmentation networks would fail to properly represent the distributive characteristic of the data and may produce unfavorable segmentation for the minority classes [43].

As pointed out in [84], the FCN-based segmentation networks simultaneously address two tasks: 1) Classification – the semantic objects should be classified correctly at the image level; 2) Localization – the classification label for a pixel should be appropriately pinpointed at the pixel level. For the classification task, the network should be invariant to the object’s location, and data need to be balanced at the image level – object classes show similar number of occurrences [101]. But for the localization task, the network should be sensitive to the object’s location, and the training data should be balanced at the pixel level – object classes show similar size [73]. However, as illustrated in Fig. 6.1, neither image-level nor pixel-level data balance is well held.
in the widely used datasets [78, 18], in multi-class semantic segmentation: At the image level, one object class may occur in more images than another; at the pixel level, one object class may show larger size than another. The goal of this research is to address both image-level and pixel-level data balancing problems in FCN-based semantic segmentation.

To balance the data, most of the FCN-based segmentation networks explore the pixel-level data balancing over the whole training dataset using the Weighted Cross-Entropy (WCE) loss [73, 92, 14, 84]. However, the WCE loss does not pay enough attention to the image-level data balancing nor the nature of network training – the networks use batch learning, and so both image-level and pixel-level class balancing should be addressed in each batch [42]. Although many conventional data balancing methods can be applied to the segmentation network [13, 24, 82], they cannot explicitly address image-level and pixel-level data balancing simultaneously. Overall, FCN-based segmentation network training using imbalanced data is still under-studied.
In this research, we propose a novel method, which is based on the WCE loss, to explicitly address image-level and pixel-level data balancing simultaneously during the FCN-based segmentation network training. Within each training batch, the image level data balancing is achieved by sampling and weighing each class based on its number of occurrences. The pixel level data balancing is achieved by weighing each class based on the object size. Furthermore, the proposed method does not change the segmentation network architecture, which can be easily applied to different baseline segmentation networks as an add-on component.

We evaluate the proposed method using two benchmark datasets in two very distinct domains: The urban street scene dataset – CityScapes [18]; The brain tumor dataset – BRATS 2015 [78]. The effectiveness of the proposed method is justified by comparing with many state-of-the-art segmentation networks [73, 92, 14, 55, 84]. The experimental results demonstrate that the proposed method can effectively improve segmentation performance. To sum up, the main contributions of this research are: 1) We propose a novel method to explicitly address image-level and pixel-level data balancing simultaneously in each training batch. 2) We demonstrate that the proposed method can be easily applied to many FCN-based semantic segmentation networks. 3) The proposed method can achieve substantially improved segmentation performance on two benchmark datasets in two different domains.

6.2 Method

The proposed method is mainly based on the WCE loss, we first discuss the limitations of the WCE loss. Then, we elaborate on the proposed image-level and pixel-level data balancing approaches. Finally, we discuss the combined image-level and pixel-level balancing approach.
6.2.1 Limitations of Weighted Cross-Entropy Loss

The FCN-based segmentation network is designed to perform image-to-image dense per-pixel predictions for semantic segmentation. Among $b$ images in the batch, let $c$ denote the segmentation class $c \in \{1, \cdots, n_c\}$, where $n_c$ is the number of classes. FCN-based segmentation network usually adopts the WCE loss:

$$
\mathcal{L} = -\frac{1}{b \cdot n_c \cdot h \cdot w} \sum_{i=1}^{b} \sum_{c=1}^{n_c} \omega_c (s_c^{(i)gt} \log s_c^{(i)pred} + (1 - s_c^{(i)gt}) \log (1 - s_c^{(i)pred}))
$$

(6.1)

where $\omega_c$ denotes the pre-defined weight of class $c$, $s_c^{(i)gt}$ and $s_c^{(i)pred}$ denote the one-hot ground truth map and the predicted map of the $i^{th}$ image of class $c$. Appropriate selection of the weights $\omega_c$ for $c \in \{1, \cdots, n_c\}$ can produce a balance in the training data at the pixel level.

By design, the WCE loss is a batch-wise loss, and the data balancing should be addressed in each batch. However, the WCE-based data balancing method only
weighs each class using a pre-defined weight, which is deemed constant throughout the whole training process. Without considering batch variations, the data in each batch is not balanced. Besides, image-level data balancing is not considered in the WCE loss.

Therefore, we adapt WCE loss by both sampling and selecting appropriate weights to achieve both image-level and pixel-level data balancing in each training batch.

### 6.2.2 Image-level Data Balancing

For each training batch, we expect the same number of occurrence for all classes, i.e., the data in the training batch is balanced at image level. For the class $c$ in the batch, there are four cases: 1) The class is balanced; 2) The class is over-represented; 3) The class is under-represented; 4) The class is not presented.

In a batch of $b$ images, let $b_c$ be the number of images with segmentation class $c$. Let $\rho_c \in [0, 1]$ denote the ratio between the number of occurrence of class $c$ and the batch size ($\rho_c = \frac{b_c}{b}$). Let $\rho_t$ denote the desired ratio between the number of occurrences of the class and the batch size. To balance a class $c$ at image level, the goal is to make $\rho_c = \rho_t$. For example, if we have a batch of size 100, with the desired $\rho_t = 0.5$, then we expect each class to occur 50 times in the batch at the image level.

In practice, for each batch, let $x^{(i)}$ denote the $i^{th}$ image, where $i \in \{1, \cdots, b\}$. We consider the following four cases to balance the data at image level, i.e., to force $\rho_c = \rho_t$, $\forall c \in \{1, \cdots, n\}$:

1. If a class $c$ is balanced, i.e., $\rho_c = \rho_t$, we set the class weight $\omega^{(i)}_{c\text{image}} = 1$, $\forall i \in \{1, \cdots, b\}$ and keep batch the same as the original batch for the class $c$.

2. If a class $c$ is over-represented, i.e., $\rho_c > \rho_t$, we have more training images than needed for this class in the batch. We can randomly select a subset of $\lceil b_c \cdot \frac{\rho_t}{\rho_c} \rceil$ images from the $b_c$ images with class $c$. In practice, we simply set $\omega^{(i)}_{c\text{image}} = 1$ if $x^{(i)}$ is selected in the sampling and set $\omega^{(i)}_{c\text{image}} = 0$ otherwise.
3. If a class $c$ is under-represented, i.e., $\rho_c < \rho_t \land \rho_c \neq 0$, we have insufficient number of training images for this class in the batch. We keep all $b_c$ images and set the class weight to be $\omega^{(i)\text{image}}_c = \frac{\rho_i}{\rho_c} > 1$, $\forall i \in \{1, \ldots, b\}$ for image-level balance.

4. If a class $c$ is not presented, i.e., $\rho_c = 0$, we do not learn for this class and simply set the weight $\omega^{(i)\text{image}}_c = 0$.

In this research, we simply select $\rho_t = 0.5$ for all classes. This way, we can balance the data at the image level.

6.2.3 Pixel-level Data Balancing

For each training image of a size $h \times w$, we expect the same number of pixels for all classes, i.e., the data in each image is balanced at pixel level. For the class $c$ in an image, there are three cases: 1) The class is balanced; 2) The class is not balanced; 3) The class not presented.

For each training image, let $p_c$ be the number of pixels of class $c$. Let $\varphi_c \in [0, 1]$ denote the ratio between the number of pixels of class $c$ and the number of pixels of the image, i.e., $\varphi_c = \frac{p_c}{h \cdot w}$. Let $\varphi_t$ denote the balanced ratio at pixel level. To balance a class $c$ at pixel level, the goal is to make $\varphi_c = \varphi_t$. For example, if we have an image of size $10 \times 10$, with 5 classes, then we expect each class to have 20 pixels.

In practice, for an input image $x^{(i)}$, we consider the following three cases to balance the data at pixel level, i.e., to force $\varphi_c = \varphi_t$, $\forall c \in \{1, \ldots, n\}$:

1. If a class $c$ is balanced, i.e., $\varphi_c = \varphi_t$, we set the class weight $\omega^{(i)\text{pixel}}_c = 1$.

2. If a class $c$ is presented but not balanced, i.e., $\varphi_c \neq \varphi_t \land \varphi_c \neq 0$, we set the class weight $\omega^{(i)\text{pixel}}_c = \frac{\varphi_t}{\varphi_c}$.

3. If a class $c$ is not presented, i.e., $\varphi_c = 0$, we do not learn for this class and set the weight $\omega^{(i)\text{pixel}}_c = 0$. 88
Figure 6.3. Examples of semantic segmentation results on CityScapes dataset. For visualization purpose, the ground truth segmentation map is superimposed to the validation image and the dashed rectangles are enlarged for highlighting improvements.

This way, the data can be balanced at the pixel level.

### 6.2.4 Combining Image-level and Pixel-level Balancing

For each training batch, we employ previously proposed the image-level and the pixel-level data balancing methods to the FCN-based segmentation network. Based on the Eq. (6.1), the WCE loss function becomes:

\[
L = -\frac{1}{2b \cdot n_c \cdot h \cdot w} \sum_{i=1}^{n_i} \sum_{c=1}^{n_c} \left( \omega^{(i)}_{\text{image}} + \omega^{(i)}_{\text{pixel}} \right) \left( s^{(i)\text{gt}} \log s^{(i)\text{pred}} + (1 - s^{(i)\text{gt}}) \log(1 - s^{(i)\text{pred}}) \right),
\]

(6.2)

where \( \omega^{(i)}_{\text{image}} \) and \( \omega^{(i)}_{\text{pixel}} \) are the weights for image-level and pixel-level data balancing, respectively. This way, we can balance the data, in each batch, at image and pixel levels simultaneously.

As illustrated in Fig. 6.2, the proposed method does not change the segmentation network architecture and only affects 1) the data selection in the batch and 2) the weights of different classes. Thus, the proposed method can be easily applied to different FCN-based segmentation networks without affecting the original network architecture.
6.3 Experiments

6.3.1 Implementation Details

The proposed method is conducted using Caffe\(^1\), PyTorch\(^2\), Tensorflow\(^3\) and Theano\(^4\) implementations with Intel Core i7 6700K, and with NVIDIA Titan X GPU with mini-batch Stochastic Gradient Descent (SGD).

The segmentation network is trained using SGD with momentum of 0.9, weight decay of 0.0001 and adaptive learning rates. We initially set the learning rate as suggested in the original papers [73, 92, 14, 55, 84]. We use the “poly” learning rate policy [14], where the initial learning rate is multiplied by \(\left(1 - \frac{\text{iter}}{\text{max iter}}\right)^{\text{power}}\) with power = 0.9. We set the batch size as suggested in the original papers. The number of overall training iterations is 100,000 for both benchmark datasets.

For fair comparisons, we do NOT apply the data balancing methods introduced in the original papers when training the networks using the proposed method. For clarification, the pre-trained FCN8s and GCN models on the CityScapes dataset are not publicly available\(^5\), even after contacting the authors. For fair comparisons, we follow the experiment setting reported in the original paper and train the model using the CityScapes dataset. However, the overall baseline performance is not as good as (FCN-8s: −2.1%) the reported performance in the original paper. For DeepLab v3, U-Net, and DeepMedic, the pre-trained models are publicly available and baseline segmentation performance in the submission is the same as the original performance.

\(^{1}\text{http://caffe.berkeleyvision.org/}\)

\(^{2}\text{https://github.com/pytorch}\)

\(^{3}\text{https://www.tensorflow.org/}\)

\(^{4}\text{http://deeplearning.net/software/theano/}\)

\(^{5}\text{https://github.com/mcordts/cityscapesScripts/issues/12}\)
Figure 6.4. Example results on segmenting brain tumor tissues: necrotic core (blue), oedema (green), non-enhancing (orange) and enhancing core (red). For visualization purpose, the dashed rectangles are used for highlighting improvements.

Figure 6.5. The segmentation performance additional gain (in percentage) over FCN-8s with WCE-based data balancing method using the CityScapes dataset. The baseline FCN-8s are trained with $\rho_t \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. 

91
Table 6.1. Average performance (in percentage) of the CityScapes testing dataset. Let ‡ denote the proposed method that balances the data for the whole dataset instead of for each batch. Compared with the baseline networks, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>Method</th>
<th>road</th>
<th>sidewalk</th>
<th>building</th>
<th>wall</th>
<th>fence</th>
<th>pole</th>
<th>traffic light</th>
<th>traffic sign</th>
<th>vegetation</th>
<th>terrain</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoveaNet</td>
<td>97.8</td>
<td>82.9</td>
<td>91.4</td>
<td>48.6</td>
<td>54.3</td>
<td>62.5</td>
<td>69.0</td>
<td>77.3</td>
<td>91.9</td>
<td>60.7</td>
</tr>
<tr>
<td>DeepLab v2</td>
<td>97.3</td>
<td>77.7</td>
<td>87.7</td>
<td>43.0</td>
<td>40.4</td>
<td>29.7</td>
<td>43.5</td>
<td>36.4</td>
<td>89.4</td>
<td>67.0</td>
</tr>
<tr>
<td>FCN-8s+Proposed‡</td>
<td>97.4</td>
<td>78.4</td>
<td>86.1</td>
<td>32.9</td>
<td>42.7</td>
<td>46.6</td>
<td>60.0</td>
<td>64.7</td>
<td>89.4</td>
<td>61.5</td>
</tr>
<tr>
<td>GCN+WCE</td>
<td>97.3</td>
<td>78.5</td>
<td>88.4</td>
<td>44.5</td>
<td>48.3</td>
<td>34.1</td>
<td>55.5</td>
<td>61.7</td>
<td>90.1</td>
<td>69.5</td>
</tr>
<tr>
<td>GCN+Proposed‡</td>
<td>97.2</td>
<td>78.5</td>
<td>88.9</td>
<td>44.5</td>
<td>48.6</td>
<td>37.1</td>
<td>57.2</td>
<td>64.8</td>
<td>90.2</td>
<td>70.9</td>
</tr>
<tr>
<td>DeepLab v3+WCE</td>
<td>98.6</td>
<td>86.2</td>
<td>93.5</td>
<td>55.2</td>
<td>63.2</td>
<td>70.0</td>
<td>77.1</td>
<td>81.3</td>
<td>93.8</td>
<td>72.3</td>
</tr>
<tr>
<td>DeepLab v3+Proposed‡</td>
<td>97.8</td>
<td>85.8</td>
<td>93.3</td>
<td>55.2</td>
<td>64.1</td>
<td>71.5</td>
<td>78.6</td>
<td>81.5</td>
<td>94.0</td>
<td>72.6</td>
</tr>
<tr>
<td>DeepLab v3+Proposed</td>
<td>98.1</td>
<td>85.9</td>
<td>93.5</td>
<td>55.3</td>
<td>64.4</td>
<td>71.9</td>
<td>79.1</td>
<td>81.9</td>
<td>94.4</td>
<td>72.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>sky</th>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>train</th>
<th>motorcycle</th>
<th>bicycle</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoveaNet</td>
<td>94.4</td>
<td>80.6</td>
<td>60.5</td>
<td>93.6</td>
<td>56.8</td>
<td>80.2</td>
<td>60.4</td>
<td>65.8</td>
<td>76.2</td>
<td>73.9</td>
</tr>
<tr>
<td>DeepLab v2</td>
<td>92.7</td>
<td>71.2</td>
<td>49.4</td>
<td>91.4</td>
<td>48.7</td>
<td>49.1</td>
<td>47.9</td>
<td>58.6</td>
<td>63.1</td>
<td></td>
</tr>
<tr>
<td>FCN-8s+Proposed‡</td>
<td>90.9</td>
<td>77.2</td>
<td>48.2</td>
<td>90.1</td>
<td>32.3</td>
<td>48.5</td>
<td>42.5</td>
<td>50.4</td>
<td>59.8</td>
<td>63.1</td>
</tr>
<tr>
<td>GCN+Proposed‡</td>
<td>90.5</td>
<td>79.2</td>
<td>49.7</td>
<td>90.8</td>
<td>34.5</td>
<td>48.8</td>
<td>44.9</td>
<td>53.2</td>
<td>61.2</td>
<td>64.6</td>
</tr>
<tr>
<td>GCN+Proposed</td>
<td>90.6</td>
<td>80.4</td>
<td>50.9</td>
<td>91.4</td>
<td>35.2</td>
<td>49.3</td>
<td>45.2</td>
<td>54.3</td>
<td>61.9</td>
<td>64.8</td>
</tr>
<tr>
<td>DeepLab v3+WCE</td>
<td>92.2</td>
<td>72.5</td>
<td>52.3</td>
<td>91.0</td>
<td>54.6</td>
<td>61.4</td>
<td>54.8</td>
<td>65.0</td>
<td>63.4</td>
<td>66.4</td>
</tr>
<tr>
<td>DeepLab v3+Proposed‡</td>
<td>92.1</td>
<td>73.1</td>
<td>54.7</td>
<td>91.1</td>
<td>55.9</td>
<td>62.1</td>
<td>53.8</td>
<td>56.6</td>
<td>66.6</td>
<td>67.6</td>
</tr>
<tr>
<td>DeepLab v3+Proposed</td>
<td>92.1</td>
<td>73.7</td>
<td>55.3</td>
<td>91.3</td>
<td>56.7</td>
<td>62.9</td>
<td>54.2</td>
<td>57.2</td>
<td>67.2</td>
<td>68.0</td>
</tr>
</tbody>
</table>

6.3.2 Comparison to Baselines and Existing Methods

To demonstrate the generalization and effectiveness of the proposed method, we train and test the proposed method in two very distinct domains using different baseline segmentation networks:

- For CityScape dataset, three baseline segmentation networks – FCN-8s [73], GCN [84] and DeepLab v3 [15] are trained, evaluated and tested.

- For BRATS 2015 dataset, two baseline segmentation networks – U-Net [22] and DeepMedic [55] are trained, evaluated and tested.

Note that the default loss function used in FCN-8s, GCN, and DeepLab v3 in the original paper is the WCE loss. We added “+WCE” for notification, which is the
Table 6.2. Average performance (in percentage) of the BRATS 2015 testing dataset. Compared with the baseline networks, for each type of tumor, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th></th>
<th>Dice</th>
<th>Precision</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net+WCE</td>
<td>82.1       66.4  60.1</td>
<td>88.2       83.9  63.8</td>
<td>81.4       60.3  62.2</td>
</tr>
<tr>
<td>U-Net+Proposed</td>
<td>84.8       68.9  63.6</td>
<td>89.7       84.0  66.7</td>
<td>85.1       64.2  68.2</td>
</tr>
<tr>
<td>DeepMedic+WCE</td>
<td>83.6       67.4  62.9</td>
<td>82.1       84.6  64.0</td>
<td>88.5       61.6  65.6</td>
</tr>
<tr>
<td>DeepMedic+Proposed</td>
<td>85.1       69.9  64.5</td>
<td>83.2       85.1  65.2</td>
<td>90.7       65.1  67.7</td>
</tr>
</tbody>
</table>

same as the original networks. For example, FCN-8s+WCE is the same as the original FCN-8s.

*Using the CityScapes dataset for evaluation*, we combine the proposed method to the baseline segmentation networks. We observe that it shows 1.8%, 1.6% and 0.7% mIoU improvements for FCN, GCN and DeepLab v3, respectively. To visually demonstrate the effectiveness of proposed method, we provide representative segmentation results of DeepLab v3 with or without the proposed method on the CityScapes dataset in Fig. 6.3. We find that the regions segmented using the proposed method are more accurate for both small and large object classes, e.g., the fences, poles and walls. The improvements are indicated by dashed rectangles, which are insufficiently segmented when using the baseline segmentation networks. The proposed method is also compared with two existing urban street scene segmentation networks: FoveaNet [66] and DeepLab v2 [14]. The quantitative results are shown in Table 6.1. Although the overall improvement of the DeepLab v3 seems limited (from 81.3% to 82.0%), this level of improvement (+0.7%) is comparable to the level of improvement achieved in the CityScape dataset in [65]. Additionally, for some object classes, e.g., fence (+1.2%), pole (+1.9%), traffic light (+2.0%), person (+1.1%), truck (+1.3%), motorcycle (+1.6%), we achieve much significant improvements. And correctly segmenting those object classes is important in the field of autonomous driving.

*Using BRATS 2015 dataset for evaluation*, the quantitative results are shown in Table 6.2. We observe that greater improvements are shown on the Sensitivity scores,
Table 6.3. Average performance (in percentage) of the CityScapes testing dataset. The baseline network is FCN-8s with WCE-based data balancing method. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th></th>
<th>road</th>
<th>sidewalk</th>
<th>building</th>
<th>wall</th>
<th>fence</th>
<th>pole</th>
<th>traffic light</th>
<th>traffic sign</th>
<th>vegetation</th>
<th>terrain</th>
<th>sky</th>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>train</th>
<th>motorcycle</th>
<th>bicycle</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>97.4</td>
<td>78.4</td>
<td>86.1</td>
<td>32.9</td>
<td>42.7</td>
<td>46.6</td>
<td>60</td>
<td>64.7</td>
<td>89.4</td>
<td>61.3</td>
<td>ρt = 0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ρt = 0.1</td>
<td>96.6</td>
<td>77.6</td>
<td>85.2</td>
<td>32.6</td>
<td>42.4</td>
<td>45.3</td>
<td>59.3</td>
<td>63.6</td>
<td>88.6</td>
<td>60.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ρt = 0.3</td>
<td>97.4</td>
<td>78.3</td>
<td>86.2</td>
<td>33.9</td>
<td>44.1</td>
<td>47.9</td>
<td>61.9</td>
<td>65.4</td>
<td>89.1</td>
<td>62.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ρt = 0.5</td>
<td>97.4</td>
<td>78.0</td>
<td>86.1</td>
<td>34.9</td>
<td>44.9</td>
<td>48.8</td>
<td>62.8</td>
<td>66.7</td>
<td>89.5</td>
<td>63.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ρt = 0.7</td>
<td>97.4</td>
<td>78.5</td>
<td>86.3</td>
<td>34.8</td>
<td>44.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
<td>63.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ρt = 0.9</td>
<td>97.4</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>44.7</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
<td>63.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

which indicates the proposed method enables the network to accurately capture more tumor tissues. Combining the proposed method to the U-Net, it shows 2.7%, 2.5% and 3.5% improvements on segmenting Com., Core and Enh., respectively. Combining the proposed method to the DeepMedic, it shows 2.5%, 2.5% and 1.6% improvements on segmenting Com., Core and Enh., respectively. To visually demonstrate the effectiveness of proposed method, we provide representative segmentation results of U-Net with or without the proposed method on BRATS 2015 dataset. As shown in the second row of Fig. 6.4, the tumor tissues, highlighted in dashed rectangle and colored in orange, are more correctly segmented when employing the proposed method.

We conclude that, by employing proposed method to the baseline segmentation networks, FCN-based segmentation networks can better capture the missing components and render more accurate segmentation results.

6.3.3 Impact of Batch-wise Data Balancing

We conduct an additional experiment (denoted using ‡) to evaluate the impact of batch-wise data balancing towards the proposed method. In this experiment, we use the proposed method to balance the whole training dataset: First, we balance the
Table 6.4. Average performance (in percentage) of the CityScapes testing dataset. The networks are trained with or without the image-level balancing (IB) method or the pixel-level balancing (PB) method. Compared with the baseline networks, for each object class, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th>Method</th>
<th>road</th>
<th>sidewalk</th>
<th>building</th>
<th>wall</th>
<th>fence</th>
<th>pole</th>
<th>traffic light</th>
<th>traffic sign</th>
<th>vegetation</th>
<th>train</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-8s+WCE</td>
<td>97.4</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>FCN-8s+IB</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>FCN-8s+PB</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>FCN-8s+Proposed</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>GCN+WCE</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>GCN+IB</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>GCN+PB</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
<tr>
<td>GCN+Proposed</td>
<td>97.3</td>
<td>78.5</td>
<td>86.3</td>
<td>34.9</td>
<td>42.7</td>
<td>48.8</td>
<td>48.8</td>
<td>62.8</td>
<td>66.5</td>
<td>89.5</td>
</tr>
</tbody>
</table>

whole training dataset at the image level with $\rho_t = 0.5$. To circumvent the dilemma of random selection induced information loss, we use image-level data balancing method to randomly generate two subsets for training. The image-level class weights are calculated based on the number of occurrences of classes at the image level for each subset. Second, we balance each subset at the pixel level. The pixel-level class weights are calculated based on total size of the class in each subset. Both image-level and pixel-level weights are considered constant during the network training. For fair comparisons, the training iterations for each subset is 50,000, i.e., total number of training iterations remains 100,000. We observe that when balancing the whole dataset using the proposed method at both image and pixel levels, the
Table 6.5. Average performance (in percentage) of the BRATS 2015 testing dataset. The networks are trained with or without the IB method or the PB method. Compared with the baseline networks, for each object class, the numbers with better performance are highlighted in blue. The numbers with the best performance are highlighted in red.

<table>
<thead>
<tr>
<th></th>
<th>Dice</th>
<th>Precision</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net+WCE</td>
<td>82.1</td>
<td>66.4</td>
<td>60.1</td>
</tr>
<tr>
<td>U-Net+IB</td>
<td>84.4</td>
<td>68.4</td>
<td>64.5</td>
</tr>
<tr>
<td>U-Net+PB</td>
<td>82.3</td>
<td>67.9</td>
<td>62.7</td>
</tr>
<tr>
<td>U-Net+Proposed</td>
<td>84.8</td>
<td>68.9</td>
<td>63.6</td>
</tr>
<tr>
<td>DeepMedic+WCE</td>
<td>83.6</td>
<td>67.4</td>
<td>62.9</td>
</tr>
<tr>
<td>DeepMedic+IB</td>
<td>83.8</td>
<td>67.9</td>
<td>63.5</td>
</tr>
<tr>
<td>DeepMedic+PB</td>
<td>85.0</td>
<td>69.8</td>
<td>64.1</td>
</tr>
<tr>
<td>DeepMedic+Prop</td>
<td>85.1</td>
<td>69.9</td>
<td>64.5</td>
</tr>
</tbody>
</table>

mIoU improvements are 1.5%, 1.2% and 0.4% for FCN-8s, GCN and DeepLab v3, respectively. Furthermore, when employing the proposed method for batch-wise data balancing, we observe that the IoU scores are further improved (on average 0.3%) for almost all classes. Thus, we can conclude that balancing the data batch-wisely is beneficial.

6.3.4 Impact of Hyper-parameter $\rho$ Selection

We further evaluate the impact of the hyper-parameter $\rho_t$ in the image-level data balancing method. We conduct experiments using different hyper-parameter $\rho_t \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for evaluation. As illustrated in Fig. 6.5, we demonstrate the additional gains when using different $\rho_t$. It is not surprising to observe that the segmentation performance drops when $\rho_t = 0.1$. This is caused by random selection induced information loss, such that the network might only see one-tenth of samples in the overall training dataset for each training epoch. By increasing $\rho_t = 0.3$, we
achieve a comparable segmentation performance to the baseline segmentation network using the WCE-based data balancing method. Further increasing the $\rho_t$ (from 0.5 to 0.9), the segmentation performance on the testing dataset first increases to a peak and then slightly drops (by 0.1%). Therefore, we simply select $\rho_t = 0.5$ for all classes.

### 6.3.5 Impact of Image-level & Pixel-level Balancing Methods

To justify the impact of the proposed image-level and the pixel-level data balancing methods, we conduct two experiments, in each of which, we remove one balancing method, and then check its impact to the final segmentation performance.

*Using the CityScapes dataset for evaluation*, the quantitative experimental results are shown in Table 6.4. When only using the image-level data balancing method, we achieve 1.2%, 1.0%, 0.3% improvements for FCN-8s, GCN and DeepLab v3, respectively. When only using the pixel-level data balancing method, we achieve 1.5%, 1.4% and 0.5% overall improvements for FCN-8s, GCN and DeepLab v3, respectively. As pointed out in Table 6.4, the segmentation performance gains of the small objects (e.g., pole, traffic light, traffic sign, rider, motorcycle, bicycle) benefit more from the pixel-level data balancing, while the segmentation performance gains of the large objects (e.g., building, wall, car, truck, bus, train) benefit more from the image-level data balancing.

*Using the BRATS 2015 dataset for evaluation*, the quantitative experimental results are shown in Table 6.5. When only using the image-level data balancing method, we achieve 2.3%, 2.0% and 4.4% Dice improvements for the U-Net and 0.2%, 0.5% and 1.4% Dice improvements for the DeepMedic on the Com., Core and Enh., respectively. When only using the pixel-level data balancing method, we achieve 0.2%, 1.5% and 2.6% Dice improvement for the U-Net and 1.4%, 2.3% and 1.2% Dice improvement for the DeepMedic on the Com., Core and Enh., respectively.

Significant improvements are achieved when using FCN-8s and U-Net as the base-
line networks. The results highlight the needs for balancing the data at the image level. According to above qualitative and quantitative results, we conclude that the proposed image-level and pixel-level balancing methods can effectively improve the performance of the multi-class semantic segmentation.

6.3.6 Impact of Batch Size Selection

To justify the impact of the batch size, we conduct experiments by using different batch size for evaluation. Specifically, when the batch size is 1, then the proposed method only performs pixel-level data balancing. Due to the fact that the image size of the CityScapes dataset is relatively large (1024 × 2048), with limited GPU memory, we only employ the DeepMedic, which inputs are the small image patches generated from the original image, to demonstrate the impact of the batch size using the BRATS 2015 dataset.

As shown in Table 6.6, by selecting batch sizes of 30, 50 (default), 70, and 90 using DeepMedic+Proposed, and the Dices for the complete tumor are 84.7%, 85.1%, 86.4%, and 86.3%, respectively, the Dices for the tumor core are 67.0%, 69.9%, 70.5%, and 70.5%, respectively, and the Dices for the enhancing tumor are 62.9%, 64.5%, 64.4%, and 64.5%, respectively. We found that the best performance is achieved when the batch size is 70.

6.4 Conclusion

We propose a novel method to batch-wisely balance training data for multi-class FCN-based segmentation networks. In the CityScape dataset, many objects like the traffic light, person, motorcycle do not always show up, and in the BRATS 2015 dataset, tumor tissues do not always show up, generating an imbalanced dataset. Correctly segmenting those object classes is important in the fields of autonomous driving and medical imaging. The proposed method addresses both image-level and
pixel-level data balancing simultaneously, and can be easily combined to many other state-of-the-art networks. By using the proposed method, we achieve substantially improved segmentation performance on two benchmark datasets – CityScapes and BRATS 2015 datasets.
7.1 Summary

In this thesis, we focus on tackling three challenges in the semantic segmentation: 1) Variation in image acquisition; 2) Variation in scale and size; 3) Variations in object size and occurrence. In the following, we summarize the findings.

In Chapter 4, we find that most state-of-the-art approaches (CNN based) seek to improve performance by fine-tuning the model using the degraded images. However, when fine-tuning the network using the degraded images, catastrophically forgetting the learned features of the clean images is inevitable. This causes an increased gap in feature distributions of higher layers. We observe that this gap in feature distributions poses a major obstacle in improving the segmentation performance of degraded images. Therefore, we address the this challenge by proposing a dense-Gram network to effectively minimize the gap in feature distributions of higher layers and improve the degrade image segmentation performance.

In Chapter 5, we find that the challenge of segmenting very small objects stands out as one of the factors behind the difference in performance. To alleviate the problems, the training data must be diverse enough to cover the scales for the objects. However, these approaches are not effective by brutal-forcedly “remembering” the object features at pre-defined scales, not to mention that these approaches would increase the training time significantly. As the spatial relationship between objects is a robust feature to reduce the scale induced bias, we propose a novel ISBMetric
to encode the global context to the network by quantifying the level of adjacency between each pair of object classes.

In Chapter 6, we find that, in multi-class semantic segmentation, at the image level, one object class may occur in more images than another; at the pixel level, one object class may show larger size than another. When presented with complex imbalanced training data, the classifier would fail to properly represent the distributive characteristic of the data and may produce unfavorable segmentation for the minority classes. However, by only re-sampling to balance one class would affect the distribution of other classes in the training set. We propose a selective-weighting method by exploiting both data sampling and class weighting to consider image- and pixel-level data balancing simultaneously for FCN-based multi-class image segmentation networks.

7.2 Future Works

7.2.1 Full Pipeline for Degraded Image Semantic Segmentation

The proposed DGN for degraded image segmentation has one potential limitation: We train a model for each degradation effect separately. As a result, the proposed method is limited to segment each degradation effect using a specific trained model. One straightforward approach addresses this limitation is to provide a degradation effect detection – to detect what degradation effect is present and then apply the pre-trained degradation effect specific model for the associated degraded image segmentation. However, in the real world, those degradation effects could simultaneously appear. For example, autonomous driving on a bumpy road while in haze day. This could cause both “random jerking” and “haze” degradation effects, not to mention that there could also be “camera noise” and “defocus”. Thus, it requires the model to tackle all degradation effects. One possible solution is to train the task (degradation effect) hierarchically – adding one degradation effect at a time while keeping the previous
performance. However, the training time will increase significantly. Efficient and effective training of such a network is also an open question.

7.2.2 LESS SUPERVISION

Deep learning methods are data-driven framework and are data-hungry – although quite powerful, it requires a large amount of data to learn the necessary representation. At the same time, with the rapid pace of data streaming, a huge amount of unlabeled/weakly-labeled data are generated everyday. The current advancement in video provides more streaming image data that could be exploited in vision learning algorithms. For example, we could consider the flow of moving objects in videos as a weak supervision for class agnostic object masks.

7.2.3 LOWER-LAYER SEMANTIC ENCODING

Current state-of-the-art semantic segmentation approaches usually concatenate the low-level layer feature maps to the high-level layer feature maps, s.t., the high-level layer can have enough boundary information and make the boundary of objects more clear. However, it is also necessary to encode the high-level layer’s semantic information into the low-level layer feature distribution. Besides, existing networks still cannot directly propagate the gradient information from deep layers to shallow layers. A better design of network gradient propagation is a potential research direction.
BIBLIOGRAPHY


[42] Emily M Hand, Carlos Castillo, and Rama Chellappa, Doing the best we can with what we have: Multi-label balancing with selective learning for attribute prediction, AAAI, 2018.


[77] Zibo Meng, Xiaochuan Fan, Xin Chen, Min Chen, and Yan Tong, *Detecting small signs from large images*, ICIRI (2017).


[80] Pauline C Ng and Steven Henikoff, *Sift: Predicting amino acid changes that affect protein function*, Nucleic acids research **31** (2003), no. 13, 3812–3814.


