Convolutional Neural Networks with Superpixels: Toward Detail-Preserving Image Segmentation

December 2021

Teppei Suzuki

A Thesis for the Degree of Ph.D. in Engineering

Convolutional Neural Networks with Superpixels: Toward Detail-Preserving Image Segmentation

December 2021

Graduate School of Science and Technology
Keio University

Teppei Suzuki

Abstract

In the computer vision field, image recognition and understanding are the main tasks. In particular, dense prediction tasks, such as image segmentation and depth estimation, are important for image editing and scene understanding. To solve such tasks, fully convolutional networks (FCNs), which is a variant of convolutional neural networks (CNNs), have been proposed and have become a de fact standard method. Although FCNs achieved better accuracy for image segmentation tasks than traditional methods, detailed information, such as image edges, boundaries, and small and/or thin objects, is often missed due to the downsampling layers, which are used for reducing computational costs and expanding receptive fields. In this thesis, the detail-preserving framework utilizing superpixels in downsampling layers is proposed. The proposed method mitigates the detailed information loss by incorporating it into existing FCNs.

Chapter 1 describes image segmentation, its application, and research questions.

Chapter 2 describes existing image segmentation methods using classical Markov random fields and deep neural networks and their variants.

Chapter 3 defines superpixel segmentation as the maximization of mutual information and then proposes an unsupervised superpixel segmentation framework using CNNs. The proposed method shows the CNNs have a strong prior for superpixel segmentation.

Chapter 4 describes graph convolutional networks and then defines convolution operations for superpixel images. Compared to general CNNs and the model replacing the convolution with the proposed convolution shows the effectiveness of superpixels in CNNs.

Chapter 5 proposes the framework implicitly incorporating a superpixel scheme into FCNs. The proposed framework demonstrates its effectiveness in various tasks and models. Moreover, the proposed framework improves the computational speeds of existing models.

Chapter 6 summarizes the results of this study and discusses future work.

Acknowledgements

本研究は，著者が慶應義塾大学大学院理工学研究科後期博士課程在学中に，同大学理工学部青木義満教授の指導のもとに行われました．本論文の執筆にあたり，多くの人からご指導，ご支援をいただいたことに感謝の意を述べさせていただきます．はじめに，本論文の主査であり，指導教員である青木義満教授には心より深く感謝申し上げます．青木義満教授には研究の指導にとどまらず，研究者としての姿勢や考え方を学ばせていただきました．また，筆者の自由奔放な研究活動に対し理解を示していただき，勉強や研究活動に関わる多くのことにご支援をいただきました．特に，社会人博士課程としての進学を快く引き受けていただき，仕事との両立をする上で多くのご支援をいただきました．学部4年生から修士課程，博士課程を通じて青木研究室で学び経験したことは今後の筆者の研究人生にとって大きな糧となることは疑う余地もありません．改めて深く感謝申し上げます．また，研究室に在籍した6年間，多くの先輩，後輩，そして同期と出会うことができました．研究室のメンバーとは研究室を卒業後も交流を続けており，彼ら，彼女らと出会えたことは一つの幸運です．

本論文の審査にあたり，快く副査を引き受けていただいた，慶應義塾大学理工学部の池原雅章教授，斎藤英雄教授，湯川正裕准教授に感謝いたします．池原雅章教授，湯川正裕准教授には，幾度も学内で研究発表をさせていただく機会があり，その度に建設的なご意見をいただきました．また，池原雅章教授とは学内にとどまらず，国内外の学会でもお会いする機会があり，研究以外のお話をさせていただくことも多々ありました．斎藤英雄教授とは，共同研究を通して研究の議論の場や国内の学会で交流させていただく機会があり，研究に対する姿勢や考え方など，勉強させていただく機会をいただきました．3人の副査の先生方に，改めて感謝申し上げます．

加えて，株式会社デンソーアイティーラボラトリの社員一同に感謝いたします．中でも，日頃から研究業務を通して議論させていただいた，佐藤育郎博士，安倍満博士，坂倉義明博士，吉田悠一様，関川雄介博士，石川康太様に深く感謝申し上げます．

最後に，私の両親に深く感謝いたします．両親には学部 4 年間と修士 2 年間，学費という形で金銭面の多大なる支援をいただきました．また，私の挑戦や決断に対し常に前向きな姿勢でいてくださったことは，研究の道を歩むことを決めた一つの要因と思います．ここに深く感謝いたします．

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

1

Introduction

In this chapter, we first introduce the motivation of image segmentation tasks. Then, we describe its applications and, finally, the research questions and contributions of this thesis.

1.1 Image Segmentation

Image recognition is the main task in the computer vision fields. Recognizing images can be achieved in several ways, such as image classification, bounding box-based detection, and image segmentation. They are widely used in intelligent systems, such as surveillance systems, automated driving systems, image retrieval, and image editing. For such tasks, image segmentation is used for the advanced processing.

For example, Figure 1.1 shows two types of example results for a road recognition task. The bounding box-based recognition roughly recognizes the road region, while segmentation-based recognition recognizes the road region in pixels. For the vision-based autonomous driving systems or the advanced driver-assistance systems, pixel-based recognition is important in accurately recognising the drivable region. For image editing, pixel-based recognition is also important to avoid editing the undesirable region.

As another example, we show the example results of person detection in Figure 1.2. For object tracking, segmentation-based detection is more useful than

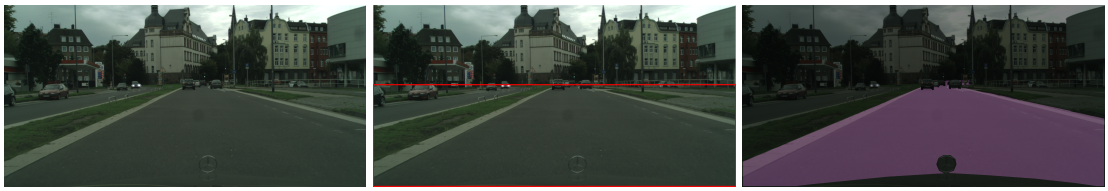


Figure 1.1: Example results of road recognition. From left to right, the input image, bounding box-based recognition, and segmentation-based recognition. The bounding box cannot represent road regions, while the segmentation mask can represent accurate road regions.

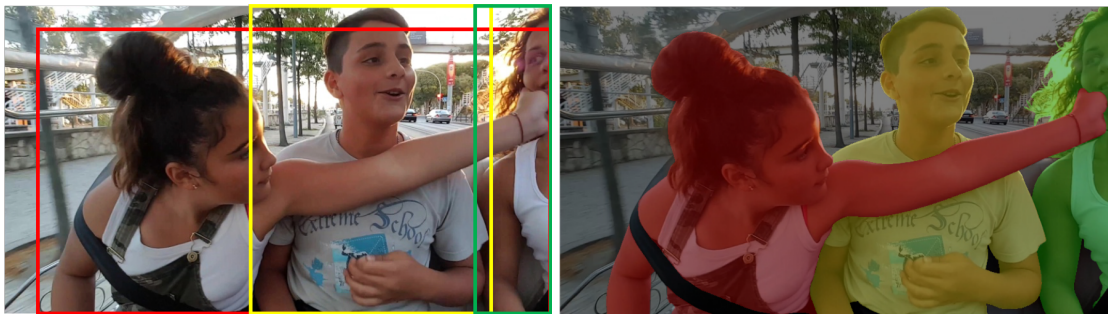


Figure 1.2: Example results of human detection. The bounding boxes contain the pixels of not only the target object but also the other objects. The segmentation masks contain only the pixels of the target object.

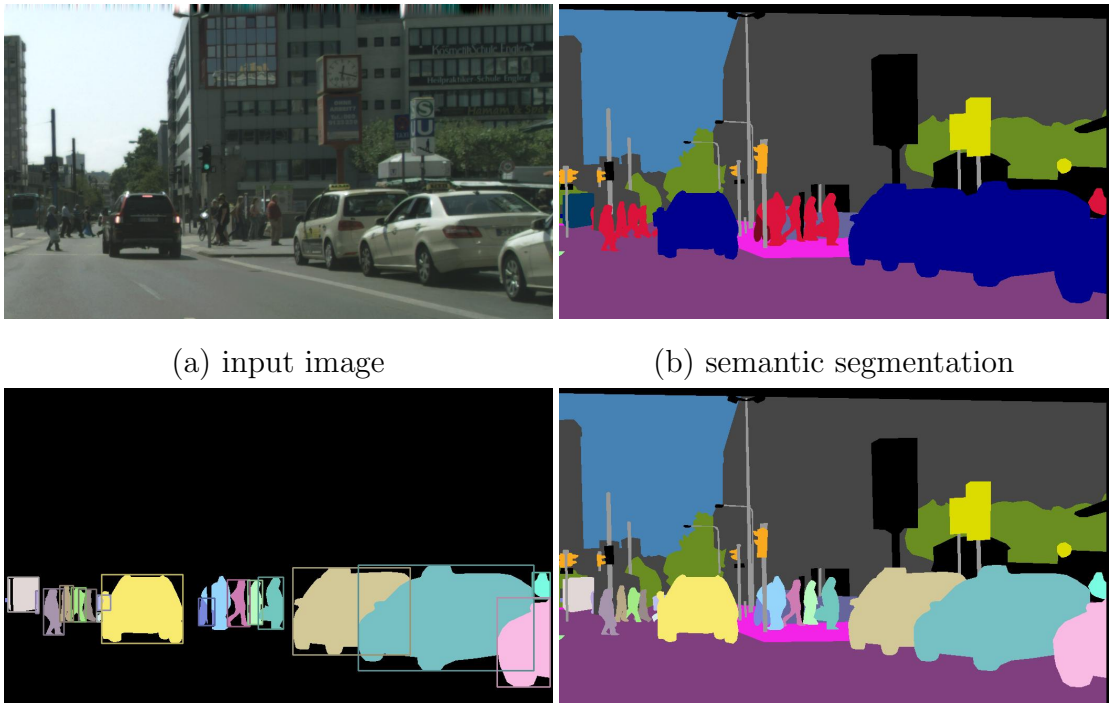
bounding box-based detection. Many object tracking algorithms track people based on the information contained in the recognized region. If one uses bounding box-based recognition, the bounding box contains not only the target objects' pixels but also other objects' pixels, as shown in Figure 1.2. Thus, the information of other objects may cause the miss assignment for the tracking problem. From these perspectives, the image segmentation may become a better choice in industrial scenes to realize the advanced systems.

1.2 Applications of Image Segmentation

Separating a foreground element of an image from its background is a fundamental application of image segmentation. Classically, chroma key, which is the placement of a blue or green screen behind the foreground to be extracted, is used for compositing digital images in filmmaking [Sawicki, 2007]. Subsequently, the foreground extraction without a blue or green screen is studied, for example, an image matting [Porter and Duff, 1984, Levin et al., 2007, He et al., 2012, Sun et al., 2004], graph cut [Boykov and Jolly, 2001, Boykov et al., 2001], and grabcut [Rother et al., 2004].

As an advanced task, semantic segmentation is a multi-region extraction task that is used for scene parsing, as shown in Figure 1.3 (b). In many cases, the regions have human-annotated labels, and semantic segmentation is solved by learning-based approaches, such as conditional random field (CRF)-based approaches [Shotton et al., 2009b, Kohli et al., 2009, Krähenbühl and Koltun, 2011] or neural network-based approaches [Long et al., 2015, Zhao et al., 2017, Chen et al., 2017a, Chen et al., 2017b, Chen et al., 2018, Yu and Koltun, 2015]. As a more advanced task, panoptic segmentation has recently been proposed by [Kirillov et al., 2019b]. Panoptic segmentation is a task combining semantic segmentation with instance segmentation, which simultaneously requires scene parsing and object detection (Figure 1.3d).

Many advanced recognition systems are built on these segmentation tasks. Some photo realistic image generation methods using a semantic segmenta-



(a) input image (b) semantic segmentation

(c) instance segmentation (d) panoptic segmentation

Figure 1.3: Illustration of segmentation tasks. (a) input image, (b) semantic segmentation (per-pixel class labels), (c) instance segmentation (per-object mask and class label, (d) panoptic segmentation (semantic segmentation + instance segmentation). This figure is taken from [Kirillov et al., 2019b].

tion map [Park et al., 2019, Johnson et al., 2018]. [Zhi et al., 2019] utilizes semantic segmentation to improve the performance of simultaneous localization and mapping (SLAM). For some object tracking benchmarks, instance masks are provided for the development of accurate object tracking systems [Voigtlaender et al., 2019, Perazzi et al., 2016, Xu et al., 2018].

1.3 Research Questions and Contributions

Fully convolutional networks (FCNs) proposed by [Long et al., 2015] have brought significant advances and have become a de facto standard method for segmentation tasks. However, detailed information, such as high frequency components and object boundaries, are missed due to downsampling operations (e.g., max/average pooling or convolution with a stride of two). Therefore, some methods are proposed to mitigate the information loss. Dilated convolution [Yu and Koltun, 2015], also known as atrous convolution [Chen et al., 2014], is used instead of striding to mitigate information loss. Trainable decoders are used for recovering lost spatial resolution [Lin et al., 2017, Ronneberger et al., 2015, Badrinarayanan et al., 2017]. These methods can generate higher quality segmentation maps than simple FCNs but require large computational costs or trainable parameters. Moreover, they do not explore better downsampling operation choices.

Thus, our research questions in this thesis are as follows:

1. *Do more effective and efficient downsampling operations exist that can help avoid the loss of the detailed information?*

2. *If such a downsampling operation exists, can it be integrated into the neural network scheme?*

As effective and efficient downsampling operations, superpixels are classically used. However, general convolutional neural networks (CNNs) cannot directly process superpixels because of their irregularity. Thus, to answer the above questions, we investigate the novel neural network architectures for superpixels. We first verify that superpixels can potentially improve a CNN's performance to answer question 1. To evaluate the effectiveness of superpixels, we propose a segmentation framework and a convolution operations for superpixels. We next propose a framework containing a superpixel segmentation scheme in the downsampling layers. We show that the method enhances the segmentation accuracy, i.e., mean intersection over union (mIoU), and/or the inference speeds of the existing architectures.

We summarize our contributions as follows:

- We propose a superpixel segmentation method based on mutual information maximization. The method optimizes a CNN for a single image in terms of the mutual information maximization objective. The method does not utilize the image prior except for local smoothness. Nevertheless, the method generates plausible superpixels and outperforms existing methods on the popular benchmarks.

- We propose convolution operations for superpixels. We identify superpixels as graphs and utilize a graph convolution scheme to process them. The method outperforms general CNNs with the same configuration on the foreground segmentation tasks.

- We propose an end-to-end image segmentation framework combining superpixels segmentation and superpixel-based downsampling. The proposed method integrates superpixel segmentation into the downsampling layers and mitigates information loss. As a result, the propose method improves the segmentation accuracy and/or inference speed of existing CNNs.

The outline of this thesis is as follows:

- *In Chapter 2*, we review the image segmentation methods. We first review the classical approaches based on the random field and then, describe the method based on neural networks, especially FCNs. Finally, after we describe the superpixel segmentation methods, including the clustering-based approach, the graph-based approach, the energy optimization-based approach, and the learning-based approach, we discuss the combination of superpixels and neural networks.

- *In Chapter 3*, we first describe the prior that CNNs have and then propose a superpixel segmentation method, which is related to the work previously published in [Suzuki, 2020, Suzuki and Aoki, 2020]. The proposed method uses CNNs, but does not require any training data. We describe the details of the loss function, which is a key component of the proposed method, and evaluate our framework in regard to the popular benchmarks for the superpixel segmentation.

- *In Chapter 4*, we first review the recent proposed graph convolutions and then describe the proposed superpixel convolution and dilated superpixel convolution, which are related to the work previously published in [Suzuki et al., 2018, Suzuki and Aoki, 2018]. Our proposed methods are provided as an extension of a general convolution operation and dilated convolution to an image domain. We compare the proposed method to general convolution using models with the same configuration.

- *In Chapter 5*, we propose the end-to-end dense prediction framework. We propose an implicit superpixel segmentation scheme that is able to be integrated into existing CNN architectures without changes in their feed-forward paths. Unlike existing methods, the proposed method does not require extra networks to generate superpixels. We show that the proposed method improves the segmentation accuracy or inference speed of the existing CNNs.

- *In Chapter 6*, we summarize all the proposed methods and evaluations in this thesis. Then, we discuss future research.

2

Image Segmentation: A Review

In this chapter, we review image segmentation methods. We fist review classical approaches using CRFs, which are still in use today and then describe recent approaches using CNNs. Finally, after we describe superpixel segmentation which is widely used in preprocessing for various image processing tasks, we discuss the combination of superpixels and CNNs.

2.1 Image Segmentation: Classical Approaches

Image segmentation has been studied for a wide range of purposes. Many tasks impose various conditions, such as image matting [Levin et al., 2007, He et al., 2012, Sun et al., 2004], using a *trimap*; graph cuts [Boykov and Jolly, 2001, Boykov et al., 2001], using *scribbles*; and the GrabCut method [Rother et al., 2004], using *bounding boxes*. In this thesis, we focus on learning-based approaches that assume that human annotated images are given. Thus, in this section, we mainly review CRF-based approaches as classical methods.

Originally, [Ren and Malik, 2003] introduced a learning-based image segmentation method, which groups pixels into superpixels and classifies each segment by a simple logistic regression classifier trained by human segmented images. Subsequently, He et al. proposed an approach to consider information about

many different object classes by introducing CRFs [He et al., 2004] and extending them with a mixture of CRFs (MoCRF) [He et al., 2006]. Let $\mathbf{X} = \{\boldsymbol{x}_i\}_{i \in \mathcal{S}}$ be the input image, where $\mathcal{S}$ denotes a set of indices of superpixels and $\boldsymbol{x}_i$ denotes the feature vector obtained from the i-th superpixel. Superpixel $\{\boldsymbol{x}_i\}$ is assigned labels $\mathbf{L} = \{\boldsymbol{l}_i\}$ from a finite label set $\mathcal{L}$. Given the context variable c from a dataset $\mathcal{C}$, MoCRF is defined as follows:

$$P(\mathbf{L}|\mathbf{X}) := \sum_{c \in \mathcal{C}} P_M(\mathbf{L}|\mathbf{X}, c) P_G(c|\mathbf{X}), \tag{2.1}$$

where $P_M(\mathbf{L}|\mathbf{X}, c)$ and $P_G(c|\mathbf{X})$ are a CRF and a gating function. $P_M(\mathbf{L}|\mathbf{X}, c)$ consists of three terms: (1) a local feature term, (2) a pair-wise feature term, and (3) a global feature term. $P_G(c|\mathbf{X})$ is a simple multi-layer perceptron. Given a set of labeled images $\mathcal{X} = \{(\mathbf{X}^n, \mathbf{L}^n\}$, the parameters of MoCRF are estimated based on the conditional maximum likelihood criterion:

$$\hat{\Theta} = \arg\max_{\Theta} \sum_{n} \log P(\mathbf{L}^n|\mathbf{X}^n), \tag{2.2}$$

where Θ denotes all parameters in the model.

Shotton et al. also proposed a CRF-based multi-class segmentation method [Shotton et al., 2006, Shotton et al., 2009a] called TextonBoost. TextonBoost's CRFs consist of shape-texture potentials, edge potentials, color potentials, and location potentials, and their parameters are estimated by a piecewise training procedure [McCallum and Sutton, 2005]. Moreover, TextonBoost utilizes textons [Leung and Malik, 2001] to capture compact representations for the range of different appearances of an object and a joint boosting algorithm [Torralba et al., 2004] as a multi-class classifier. Kohli et al. proposed Robust P^n CRFs using higher order potentials and improved the segmentation accuracy around object boundaries, although it is not a learning-based method.

Krähenbühl and Koltun proposed fully connected CRFs for the original pixel space [Krähenbühl and Koltun, 2011]. They realize the efficient inference in fully connected CRFs by a mean field approximation. Let $\mathbf{X}$ and $\mathcal{L}$ be a random field defined over a set of variables $\{X_1, \ldots, X_N\}$ and a set of possible labels $\{l_1, \ldots, l_k\}$. Also, we define a random field $\mathbf{I}$ over variables $\{I_1, \ldots, I_N\}$. A CRF

$(\mathbf{X}, \mathbf{I})$ is characterized by a Gibbs distribution $P(\mathbf{X}|\mathbf{I})$. Given the complete graph $\mathcal{G}$ on the image $\mathbf{X}$, the Gibbs energy of a labeling $\mathbf{x} \in \mathcal{L}^N$ is defined as follows:

$$E(\mathbf{x}) = \sum_i \underbrace{\psi_u(x_i)}_{\text{unary potential}} + \sum_{i<j} \underbrace{\psi_p(x_i, x_j)}_{\text{pair-wise potential}} , \tag{2.3}$$

where i and j range from 1 to N. The unary potential is computed as a classifier, and the pair-wise potential is defined as follows:

$$\phi_p(x_i, x_j) := \mu(x_i, x_j) \sum_{m=1}^{K} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j), \tag{2.4}$$

where $k^{(m)}$ denotes a Gaussian kernel, $\mathbf{f}_i$ and $\mathbf{f}_j$ are feature vectors for pixel i and j, $w^{(m)}$ are linear combination weights, and μ is a label compatibility function. Instead of an exact distribution P, Krähenbühl and Koltun compute a distribution $Q(\mathbf{X})$ that minimizes the KL divergence $\mathbf{D}(Q||P)$ among all distributions Q that can be expressed as a product of independent marginals, $Q(\mathbf{X}) = \prod_i Q_i(\mathbf{X}_i)$. Then, the following iterative update equation is derived:

$$Q_i(x_i = l) = \frac{1}{Z_i} \exp\left\{ -\psi_u(x_i) - \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^{K} w^{(m)} \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l') \right\}. \tag{2.5}$$

This mean field approximations enable the computation of a fully connected CRFs on a graph that has billions of edges.

Figure 2.1 shows example results of various CRF-based methods. The robust P^n CRFs [Kohli et al., 2009] generate a better segmentation result than the grid CRFs [Shotton et al., 2009a] because of the higher order potentials. The fully connected CRFs fit the high-frequency components because of the dense connections of CRFs.

2.2 Image Segmentation in A Neural Network Era

In 2012, Alex et al. proposed AlexNet using CNNs for performing large-scale image classification tasks [Krizhevsky et al., 2017] and demonstrating their effectiveness. Many researches were inspired their work, and then, CNNs have become

Figure 2.1: Example results of various CRF-based semantic segmentation methods. From left to right, an input image, grid CRFs [Shotton et al., 2009a], robust P^n CRFs [Kohli et al., 2009], fully connected CRFs [Krähenbühl and Koltun, 2011], and ground truth. The results are taken from [Krähenbühl and Koltun, 2011].

essential tools in some research fields, such as computer vision, natural language processing, and signal processing. Image segmentation was also affected by the success of AlexNet.

2.2.1 Early Work for Image Segmentation Using Neural Networks

In the case of image segmentation tasks, patch-based classification models have been proposed [Mnih and Hinton, 2010, Mnih and Hinton, 2012, Ciresan et al., 2012, Pinheiro and Collobert, 2014, Saito et al., 2016], which divide an image into patches, classify each patch, and identify the predicted label as the label of the centroid pixel of the patch.

Farabet et al. proposed a method using hierarchical features obtained by CNNs [Farabet et al., 2012]. The method combines the hierarchical features and superpixels and predicts semantic labels from raw pixels by CRFs.

These methods have demonstrated good results on several segmentation tasks, but they have some issues. The patch-based approaches accept only a fixed input size of images. Thus, one needs to preprocess an input image, such as

resizing, and it may generate undesirable artifacts. The method proposed by [Farabet et al., 2012] does not require such preprocessing, but their system flow is slightly complex. As a result, FCNs [Long et al., 2015] (introduced in the next section) have been proposed and have become the *de fact* standard for dense prediction tasks, such as image segmentation.

2.2.2 Fully Convolutional Neural Networks

Long et al. proposed FCNs [Long et al., 2015] for semantic segmentation tasks, and other various derivative models have been proposed [Chen et al., 2014, Chen et al., 2017b, Chen et al., 2018, Lin et al., 2017, Ronneberger et al., 2015, Yu and Koltun, 2015]. FCNs consist of only convolutional layers with nonlinear activation functions, and arbitrary-sized inputs can be fed into FCNs because convolution operations do not depend on the input size. Thus, the training protocol of FCNs is simple, but FCNs show better segmentation accuracy than other methods [Gupta et al., 2013, Farabet et al., 2012, Pinheiro and Collobert, 2014]. As a results, the applications of FCNs have been extended to various tasks, such as depth estimation [Fu et al., 2018, Godard et al., 2017, Godard et al., 2019], optical flow estimation [Dosovitskiy et al., 2015, Ilg et al., 2017, Sun et al., 2018], superpixel segmentation [Jampani et al., 2018, Suzuki, 2020, Tu et al., 2018, Yang et al., 2020, Suzuki, 2021], and fundamental inverse problems (e.g., deblurring [Nah et al., 2017, Tao et al., 2018] and inpainting [Iizuka et al., 2017, Yeh et al., 2017]).

Long et al. proposed three types of FCNs: FCN-8s, FCN-16s, and FCN-32s. The number at the end denotes the output stride. For example, the FCN-32s generates a segmentation map with a reduced spatial resolution of 1/32 of the input size. Because the FCN-32s uses bilinear interpolation, which is a static and linear interpolation method, to recover the lost spatial resolution, but local structures such as object boundaries are often missed. Although FCN-16s and FCN-8s mitigate the issue by aggregating the predictions generated from high-resolution intermediate feature maps, the issue still remain, as shown in Figure 2.2. Moreover, this issue may be caused by the methods that generate the prediction map from coarse outputs by utilizing bilinear interpolation. Therefore,

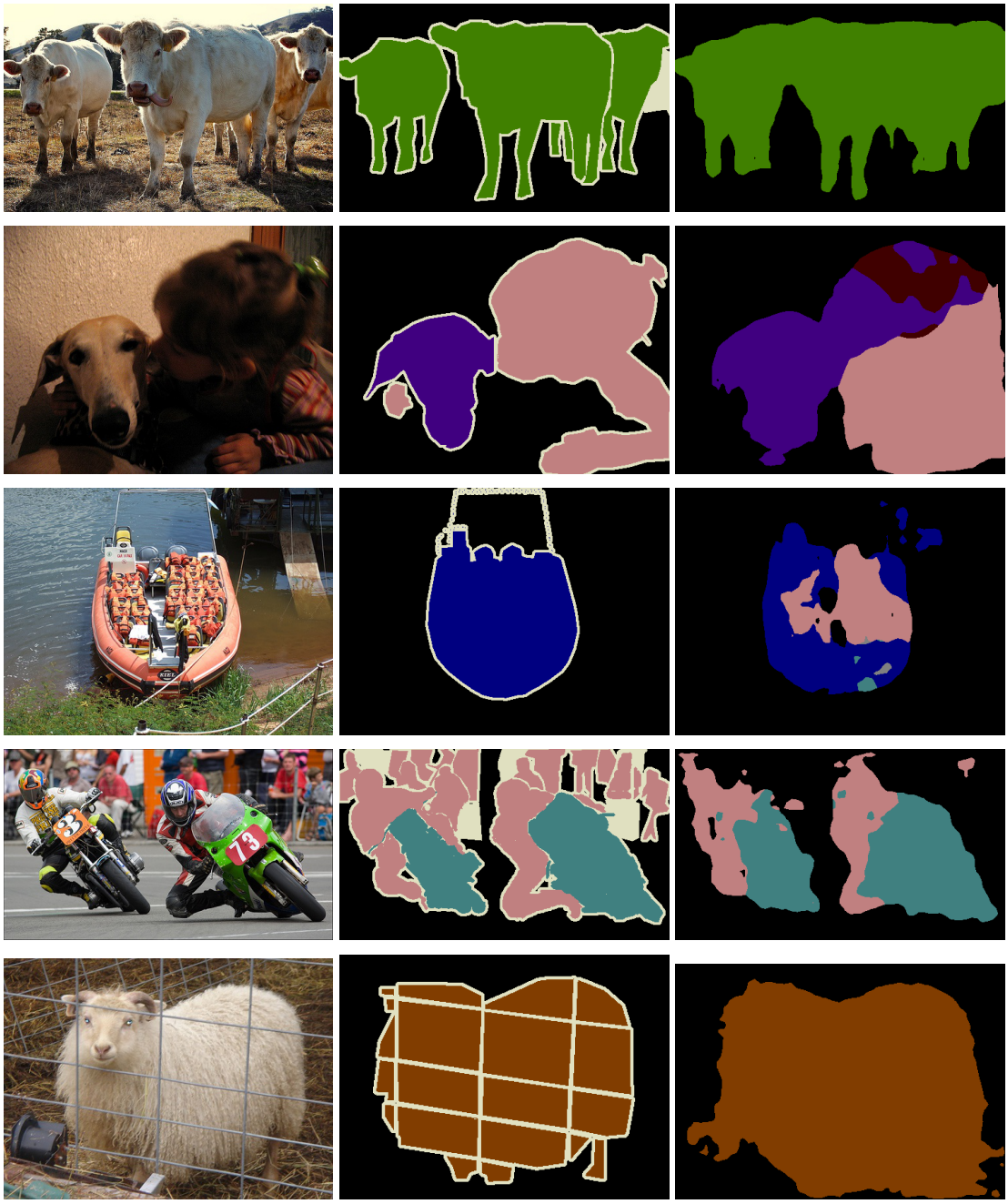


Figure 2.2: Example results of FCN-8s. From left to right, the input image, the ground truth, and the prediction. The predictions miss accurate object boundaries. These results are taken from [Long et al., 2015].

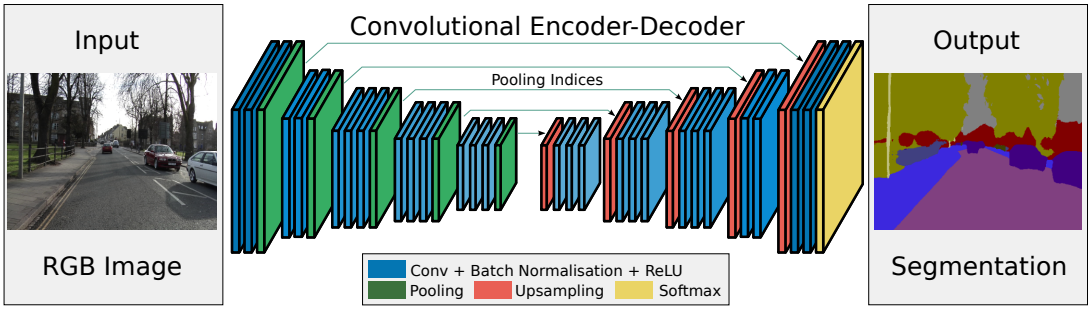
to generate a high-resolution map, many existing methods utilize a trainable decoder or replace the striding with atrous convolution, also known as dilated convolution [Chen et al., 2014, Yu and Koltun, 2015].

The encoder-decoder models [Badrinarayanan et al., 2017, Lin et al., 2017, Noh et al., 2015, Ronneberger et al., 2015, Li et al., 2020] recover the lost resolution and local structure using a trainable decoder that consists of transposed convolutions or bilinear interpolations with some convolution layers. SegNet [Badrinarayanan et al., 2017] uses a trainable decoder that has an architecture symmetric to its encoder (Figure 2.3a). Moreover, SegNet uses sampled indices in the downsampling layers in the encoder to upsample low-resolution maps. SegNet can recover the detailed structures in images, but recovered maps are sparse. Thus, the decoder needs the same number of convolution layers as the encoder to fill holes. To avoid it, transposed convolutions, which recover the lost resolution by trainable filters, are often used.

SegNet and some similar architectures [Noh et al., 2015, Kendall et al., 2015] use only low level information, such as pooling indices, and/or depend on the learned convolution to recover the resolution. Therefore, the decoder often leads to wrong structures. U-Net [Ronneberger et al., 2015] utilizes not only downsampled feature maps but also feature maps before downsampling by skip connections to recover the resolution (Figure 2.3b). Because of the skip connections, U-Net can recover detailed structures, and it is widely used in various tasks, such as image generation [Schonfeld et al., 2020] and image editing [Isola et al., 2017]. Such encoder-decoder models can generate a high-resolution map but require additional layers and parameters for the decoder.

Another approach to mitigate information loss due to downsampling is atrous convolution [Chen et al., 2014, Yu and Koltun, 2015], also known as dilated convolution [Yu and Koltun, 2015]. It efficiently expands receptive fields instead of striding. Many modern architectures, such as PSPNet [Zhao et al., 2017] and DeepLabv3 [Chen et al., 2017b], use atrous convolution and generally generate a prediction map with an output stride of eight, meaning the resolution of the predicted map is eight times smaller than that of the input image. Although such models demonstrate effective results, they typically require high computational

(a)



(b)

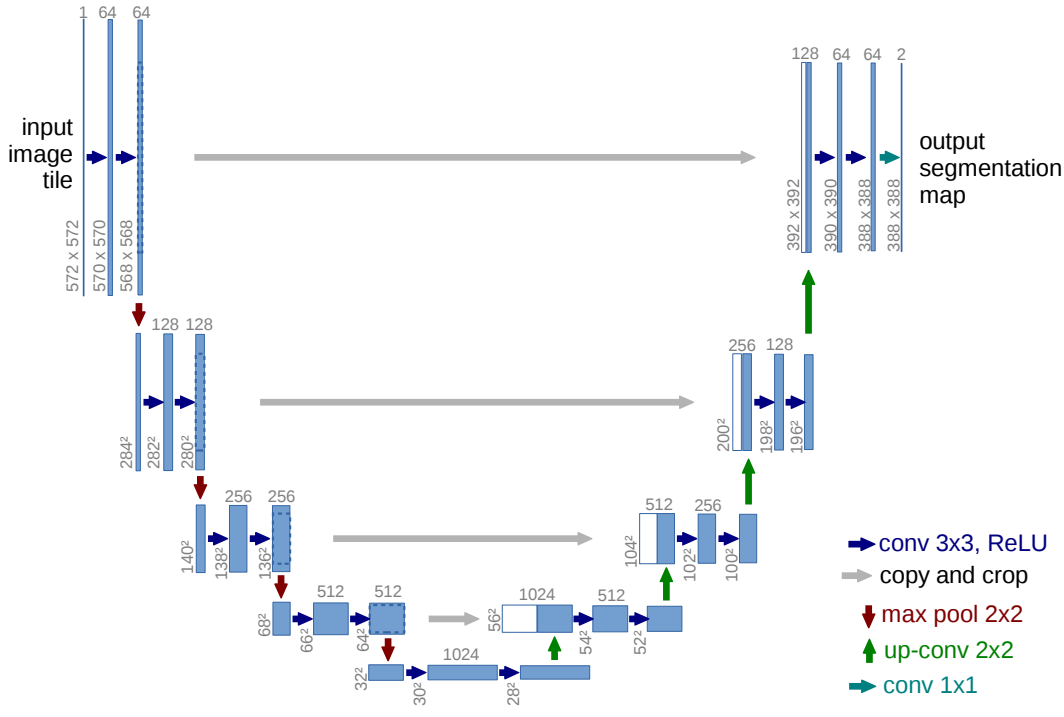


Figure 2.3: Illustration of (a) SegNet [Badrinarayanan et al., 2017] and (b) U-Net [Ronneberger et al., 2015]. These figures are taken from [Badrinarayanan et al., 2017] and [Ronneberger et al., 2015], respectively.

costs because they discard some downsampling operations and process a large number of pixels in intermediate layers.

Deeplab [Chen et al., 2017a] and CRF as RNN [Zheng et al., 2015] utilize fully connected CRFs [Krähenbühl and Koltun, 2011] for postprocessing to refine the segmentation results. In particular, [Zheng et al., 2015] realize end-to-end training and simplify the system flow by defining CRFs as recurrent neural networks. CRFs and other techniques, including atrous convolution, decoder, and our proposed method in Chapter 5, are compatible, and their combination may produce even better results.

2.3 Superpixel Segmentation

As a detail-preserving complexity reduction process for image data, superpixel segmentation is classically used [Ren and Malik, 2003, He et al., 2006, Uijlings et al., 2013, Matsuo and Aoki, 2015, Takayama et al., 2016]; it groups pixels similar in color and other low-level properties (Figure 2.4). Superpixels can reduce the complexity of image data while preserving object boundaries and semantics. Some existing neural network architectures utilize superpixels [Gadde et al., 2016, Knyazev et al., 2019, Kwak et al., 2017, Suzuki et al., 2018, Yang et al., 2020, Zhang et al., 2019]. Superpixels were originally proposed by [Ren and Malik, 2003] and were realized by the normalized cut algorithm [Shi and Malik, 2000]. In this section, we categorize superpixel segmentation methods into four types and review them.

2.3.1 Clustering-based Method

SLIC [Achanta et al., 2012] is a simple method based on k-means clustering in the *labxy* color-image plane space. Let $(l, a, b)^\top$ and $(x, y)^\top$ be the CIELAB space and pixel's position. SLIC initializes k cluster centers as the centroids of the regular grid as shown in Figure 2.5. Many other superpixel segmentation approaches adopt this initialization procedure. Then, the distance function for

Figure 2.4: Example results of superpixels. The results are taken form [Achanta et al., 2012].

Figure 2.5: Illustration of centroid initialization in SLIC when the number of initial superpixels is 192. The left image denotes an input image, and the right image denotes the boundary of initial superpixels. The centroid has the mean position and color of the pixels contained in each grid.

k-means clustering in SLIC is defined as follows:

$$D := \sqrt{d_c^2 + \left(\frac{d_s^2}{S}\right) m^2}, \tag{2.6}$$

where d_c and d_s denote the Euclidean distance in the CIELAB space and pixel position space. S is the grid interval of the initial regular grid, and m is a weight parameter to balance the importance between color similarity and spatial proximity. Moreover, [Achanta et al., 2012] extend SLIC to supervoxels by introducing the z-coordinate.

SLIC is a popular method that is widely used because of its simple algorithm. There are many derivative methods; manifold SLIC [Liu et al., 2016], which maps an image into a 2-dimensional manifold $\mathcal{M} \subset \mathbb{R}^5$ and utilizes a better distance measure. DASP [Weikersdorfer et al., 2012] utilizes depth as additional information to generate better superpixels.

Clustering-based methods are simple but effective methods. However, they do not guarantee the connectivity of pixels assigned to the same superpixel. Therefore, the methods generally require postprocessing to enforce the connectivity.

2.3.2 Graph-based Method

Graph-based methods formulate superpixel segmentation as graph partitioning [Grady, 2006, Liu et al., 2011, Veksler et al., 2010]. Normalizing cut [Shi and Malik, 2000] is a popular graph-based method, which is especially used in CRF-based segmentation [Ren and Malik, 2003, He et al., 2006]. Normalizing cut produces regular, visually pleasing superpixels, but it requires $O(N^1.5)$ complexity.

Felzenswalb and Huttenlocher [Felzenszwalb and Huttenlocher, 2004] proposed a more efficient graph-based method. Let $G = (V, E)$ be a graph on an image with n vertices and m edges, and $S = (C_1, \ldots, C_r)$ be a set of segments, where C_r denotes a set of the vertices. Then, their algorithm is represented as follows:

1. Sort E into $\pi = (o_1, \ldots, o_m)$ by non-decreasing edge weight.
2. Start with segmentation S^0, where each vertex v_i is in its own component.
3. Repeat step 4 for $q = 1, \ldots, m$.
4. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering; that is $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components; otherwise, do nothing.
5. Return $S = S^m$.

The algorithm runs in $O(m \log m)$, which is more efficiently than normalizing cut.

2.3.3 Energy Optimization

Superpixel segmentation based on energy optimization iteratively optimizes a formulated energy [Mester et al., 2011, Van den Bergh et al., 2012, Tasli et al., 2013, Yao et al., 2015]. For example, SEEDS [Van den Bergh et al., 2012] defines the energy function consisting of a color distribution term and a boundary term. The energy function of

ETPS [Yao et al., 2015] consists of five terms, a shape regularization term, an appearance coherence term, a boundary length term, a topology preservation term, and a minimum size term. Because of the topology preservation term defined as Dirac's delta function, ETPS preserves pixel connectivity and does not require postprocessing to enforce the connectivity. Thus, ETPS generates high-quality superpixels in realtime.

As a heterogeneous approach, Kanezaki proposed a CNN-based method [Kanezaki, 2018]. Kanezaki's method is also classified into the energy optimization method, which optimizes the CNN's parameters to minimize the energy function. The method cannot control the number of segment; hence, strictly speaking, it is not superpixel segmentation. The method proposed in Chapter 3 is inspired by the method and its insight.

2.3.4 Learning-based Method

Recently, learning-based superpixel segmentation methods have been proposed. Learning-based superpixel segmentation has several challenges: (1) no ground truth exists for superpixels, (2) the indices of different superpixels are interchangeable, and (3) existing superpixel segmentation methods are not differentiable.

Tu et al. [Tu et al., 2018] attacked these challenges and proposed SEAL. SEAL learns pixel affinities using a proposed segmentation-aware loss. Given the learned CNNs, SEAL generates superpixels using ERS [Liu et al., 2011] with the affinity generated CNNs. SEAL significantly improves the quality of superpixels, but the training procedure is not superpixel aware. Moreover, SEAL is not an end-to-end framework; hence, computational overheads are inevitable.

[Jampani et al., 2018] proposed an end-to-end superpixel segmentation framework, called the superpixel sampling network (SSN). Jampani et al. made the SLIC procedure differentiable and established the end-to-end framework. Let $I \in \mathbb{R}^{n \times 5}$ be the CIELAB image with pixel positoins, where n denotes the number of pixels. Then, the SSN extracts k-dimensional pixel features with CNNs, $\mathcal{F} : I \to F \in \mathbb{R}^{n \times k}$ and generates soft superpixel associations $Q \in \mathbb{R}^{n \times m}$ with the following differentiable SLIC.

1. Initialize superpixel centers $S^0 \in \mathbb{R}$ with the same procedure as SLIC.

2. Repeat step 3 for $t = 1, \ldots, v$.

3. Compute the assignment between each pixel p and the surrounding superpixel i, $Q_{pi}^t = \exp(-||F_p - S_i^{t-1}||^2)$. Compute new superpixel centers, $S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^n Q_{pi}^t F_p$; $Z_i^t = \sum_p Q_{pi}^t$.

Optionally, the SSN computes hard-associations $H_p^v = \underset{i \in \{0,\ldots,m-1\}}{\arg\max}\ Q_{pi}^v$ and enforces the connectivity in the same way as SLIC. To train the CNNs, Jampani et al. also proposed the task-specific reconstruction loss, which is defined as $\mathcal{L}(R, \tilde{Q}\hat{Q}^\top R)$, where $R \in \mathbb{R}^{n \times l}$ denotes the pixel-wise ground truth, and $\tilde{Q}$ and $\hat{Q}$ are row-wise and column-wise normalized matrix of Q.

[Yang et al., 2020] proposed an end-to-end one-shot framework, which improves the computational overheads of SSN. SSN requires an iterative procedure for differentiable SLIC, and Yang et al. pointed out its computational overheads. Their proposed method defines superpixel segmentation as a classification problem. The method directly predicts the soft assignment matrix Q through the softmax function. The method is trained with the similar reconstruction loss to the SSN. Yang et al. combined the proposed superpixel segmentation network with PSMNet, which is the depth prediction network; it shows that the effectiveness of end-to-end training on the depth estimation task.

Superpixels were developed as preprocessing for the image processing; hence, one think that integrating superpixels into CNNs may be slightly uncomfortable. However, we exploits only the concept of the superpixels, i.e., the detail-preserving pooling. The proposed method introduced in Chapter 5 does not use superpixels as downsampling, but as data representation. The downsampled feature maps have grid structures, but grid cells correspond to superpixels. As a results, the proposed method can realize the detail-preserving image segmentation in the general CNNs.

2.4 Convolutional Neural Networks with Superpixels

Combining CNNs with superpixels comes with some challenges. The reason is the irregularity of superpixels, meaning that the pixels on images are aligned on a regular grid, but superpixels are not. [Yang et al., 2020] combined them by assuming that superpixels are aligned on a regular grid and showed good results. However, the two networks work independently; one is for superpixel segmentation, and the other is for the task of interest. Bronstein et al. proposed the graph convolution and utilized it for superpixel-based handwritten digit recognition [Monti et al., 2017]. Graph convolution is one way to combine superpixels with neural networks, but existing CNN models do not utilize it the same way.

The goal of this thesis is to integrate a superpixel segmentation scheme into general CNN frameworks without additional segmentation networks. We first investigate the prior of CNNs for superpixel segmentation in Chapter 4 and then the effectiveness of superpixel-based segmentation in Chapter 3. Finally, we propose the integrated framework in Chapter 5.

Unlike detail-preserving pooling [Saeedan et al., 2018], the proposed method focuses on the upsampling, i.e., the proposed method preserves the detailed information to upsample the low-resolution outputs. Thus, the aim of the proposed method is similar to the trainable decoder. However, the detail-preserving pooling preserves the information to recognize and understand the image. In summary, the detail-preserving pooling is for image-to-vector tasks, such as image classification, and the proposed method for image-to-image tasks, such as dense prediction tasks.

3

Superpixel Segmentation via Convolutional Neural Networks with a DIP framework

Superpixels are low-dimensional representations for images and generally given as a set of pixels similar in color and other low-level properties (e.g., SLIC [Achanta et al., 2012], SEEDS [Van den Bergh et al., 2012], ETPS [Yao et al., 2015], and Felzenswalb and Huttenlocher's method (FH) [Felzenszwalb and Huttenlocher, 2004]). Superpixel segmentation is generally used as preprocessing for image processing tasks.

Many existing methods depend on local and low-level properties, such as local connectivity, color, and positions, to generate superpixels. If the number of superpixels is large, the methods using the properties work well because the images almost consist of low-frequency components and locally consist of pixels having the same color. However, if the number of superpixels is small, superpixels need to group a wide range of pixels with various properties, and it is a difficult task for the methods using the local and low-level properties. Figure 3.1 shows the result in such a case.

To introduce non-local properties into superpixel segmentation, we propose a CNN-based superpixel segmentation method. According to

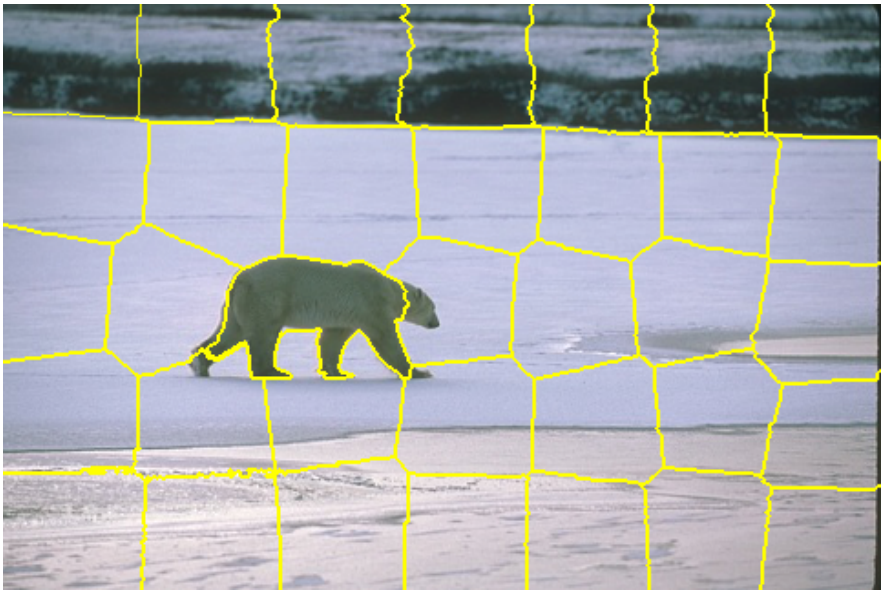


Figure 3.1: Example result of ETPS [Yao et al., 2015] with 25 superpixels. Parts of the object and background pixels are grouped into the same superpixel.

[Ulyanov et al., 2018], CNNs have a prior for images even though it is not trained. In fact, CNNs can produce much cleaner results with sharper edges for inverse problems than methods using the hand-crafted prior. To leverage the CNNs' prior, we utilize the same procedure as deep image prior (DIP) [Ulyanov et al., 2018], which optimizes a randomly-initialized CNN using a single image in inference time without labels. We assume that the prior verified by [Ulyanov et al., 2018] also works well for superpixel segmentation, especially for capturing a global structure. If the prior works as expected, our CNN-based method should have better performance than other methods with a small number of superpixels.

In the rest of this chapter, we firstly introduce DIP, which was demonstrated by [Ulyanov et al., 2018], and then we propose our CNN-based superpixel segmentation method. Finally, we evaluate our proposed method on the popular superpixel segmentation benchmarks.

3.1 Image Prior of Convolutional Neural Networks

CNNs have achieved remarkable results for several computer vision tasks, as already described. We believe that the achievement is a result of not only the large amount of data and general-purpose computing on graphics processing units (GPGPU) but also a CNNs' prior.

Indeed, [Ulyanov et al., 2018] showed that CNNs have a strong prior for image data. They prove it with standard inverse problems, such as denoising, super-resolution, and image restoration. The inverse problems result in having to reproduce the original signal from the observed signal.

Classically, the inverse problems are basically solved by energy minimization problems; for example, the denoising problem can be solved as the least squares problem as follows:

$$\min_{\hat{I}} ||I - \hat{I}||^2, \tag{3.1}$$

where I and $\hat{I}$ are the observed image and the variable, respectively; $||\cdot||^2$ denotes squared $L2$-norm. The solution of eq. (3.1) is $\hat{I} = I$, and it completely reconstructs the observed image. Thus, eq. (3.1) is generally introduced a *prior term* as a regularizer as follows:

$$\min_{\hat{I}} ||I - \hat{I}||^2 + \lambda R(\hat{I}), \tag{3.2}$$

where $R(\cdot)$ indicates the prior term, and λ is the coefficient. For the image data, the prior $R(\cdot)$ may be the total variation,

$$TV(\hat{I}) := \sum_p \partial_x \hat{I} + \sum_p \partial_y \hat{I}, \tag{3.3}$$

where $\partial_{\{x,y\}}$ denote the image gradient. Note that $\sum_p$ denotes the sum of all the pixels of the gradient image.

As seen in eq. (3.2), the prior term is necessary to predict the original signal from the observed signal because the observed signal is reconstructed if there is no prior term. However, Ulyanov et al. [Ulyanov et al., 2018] showed that the

(a) (b) (c)

Figure 3.2: Example results of denoising (top) and super-resolution (bottom) by DIP [Ulyanov et al., 2018]; (a), (b), and (c) denote the original image, observed image (the results of bilinear interpolation for super-resolution), and the DIP result. The results are taken from [Ulyanov et al., 2018].

randomly initialized CNNs can predict the original image without training data by optimizing its parameters with reference to the following objective function:

$$\min_{\theta} ||I - f_\theta(z)||^2, \tag{3.4}$$

where $f_\theta(\cdot)$ denotes the CNNs parameterized θ, and z denotes the fixed random vector whose dimension is generally lower than that of the observation image. Example results of denoising and super-resolution by DIP are shown in Figure 3.2.

Ulyanov et al. considered that the unreasonable fact is caused by the prior

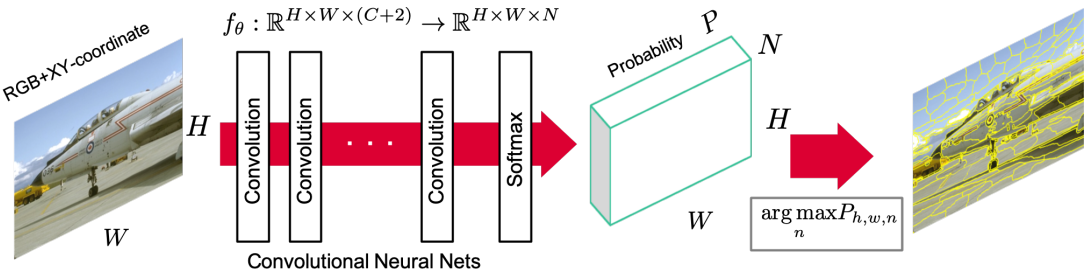


Figure 3.3: Overview of our proposed superpixel segmentation method. We define superpixel segmentation as the N-class classification task. The parameters of CNNs are optimized to minimize eq. (3.6) for a single image.

of the CNNs, which they refer to as DIP. In other words, the CNNs can capture the image structure without training from a large amount of data. Therefore, to generate superpixels using the global information and image structure information, such as object and background structure, we propose CNN-based superpixel segmentation method that utilizes the prior.

3.2 Superpixel Segmentation via Convolutional Neural Networks

Our proposed method is built on the DIP framework; the method receives an image as input and optimizes a randomly initialized CNN. However, the DIP framework aims to solve inverse problems, and superpixel segmentation is out of scope. Therefore, we design an objective function to execute superpixel segmentation with the DIP procedure. The proposed method defines superpixel segmentation as an N-class classification task, and our proposed objective function is based on the mutual information. We show an overview in Figure 3.3.

Let $I \in \mathbb{R}^{H\times W\times C}$ be an input image, where H, W, and C denote image height, width, and input channels (typically RGB), respectively. Our goal is to assign superpixels $\mathcal{S} = \{s_1, \ldots, s_N\}$ to all pixels, where s_n denotes the n-th superpixel to be a set of pixels, and the number of superpixels N is a hyperparemeter.

Note that the proposed method allows s_n to be an empty set. Therefore, the hyperparameter N is the upper bound of the number of superpixels.

Let $P \in \mathbb{R}_+^{H\times W\times N}$; $\sum_n P_{h,w,n} = 1$ be a probabilistic representation of superpixels, where $\mathbb{R}_+$ indicates non-negative real number. The superpixel assigned to a pixel at (h, w) is as follows:

$$s_k := \left\{ x_{h,w} \;\middle|\; k = \arg\max_n P_{h,w,n} \right\}, \tag{3.5}$$

where $x_{h,w}$ denotes the pixel as (h, w). We obtain P from an input through the CNNs, $f_\theta : I \to P$, where θ denotes the CNNs' parameters. The parameters are optimized to minimize the following objective function:

$$\mathcal{L}_{\text{objective}} := \mathcal{L}_{\text{mi}} + \alpha\mathcal{L}_{\text{smooth}} + \beta\mathcal{L}_{\text{recons}}, \tag{3.6}$$

where $\mathcal{L}_{\text{mi}}$, $\mathcal{L}_{\text{smooth}}$, and $\mathcal{L}_{recons}$ denote a mutual information term for unsupervised, respectively; α and β are hyperparemeters balancing the importance of each term.

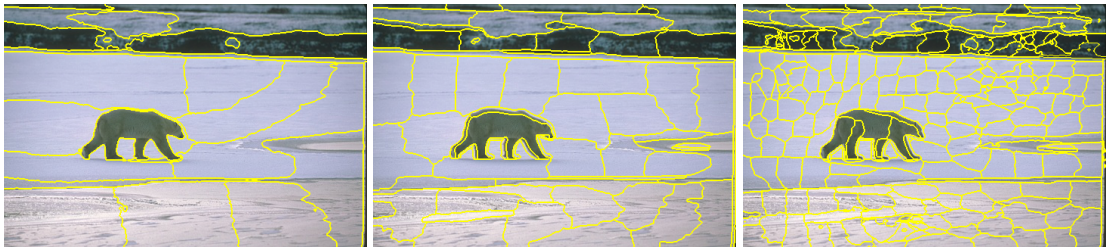
3.2.1 Mutual Information

$\mathcal{L}_{\text{mi}}$ is a mutual information term between an input image and the probability, which was originally proposed by [Bridle et al., 1992]. $\mathcal{L}_{\text{mi}}$ is as follows:

$$\mathcal{L}_{\text{mi}} := \frac{1}{HW}\sum_{h,w}\sum_{n} -P_{h,w,n}\log P_{h,w,n} + \lambda\sum_n \hat{P}_n \log \hat{P}_n, \tag{3.7}$$

where $\hat{P} \in \mathbb{R}^N$, $\hat{P}_n = \frac{1}{HW}\sum_{h,w} P_{h,w,n}$ denotes the mean value of the probability vectors over all the pixels. The first term is the mean of the entropy of $P_{h,w} \in \mathbb{R}^N$ over all the pixels, and the minimization of it encourages deterministic superpixel assignment. The second term is the negative entropy of the mean vector over all the probability vectors, and the minimization of it encourages the size of each superpixel to be uniform.

Unlike the mutual information proposed by [Bridle et al., 1992], we introduce a scalar value λ as a coefficient of the second term to control the number of superpixels. When λ is small, the model tends to try to segment an image with



$\lambda = 0$ $\lambda = 1$ $\lambda = 1.5$

Figure 3.4: Example results of the proposed method with various λ values. When λ increases, the number of superpixels also increases.

a small number of superpixels because the first term becomes dominant, and the model ignores the second term. As λ increases, the number of superpixels given by the CNN converges on N. However, if λ is too large, $P_{h,w}$ tends to become uniform because the second term becomes dominant and the uniform probability is a local minimum. In practice, the proposed method works well when λ is within $[0, 3]$. We show example results with various λ values in Figure 3.4.

3.2.2 Spatial Smoothness

The smoothness term is a basic prior for inverse problems in image processing tasks, which quantifies the difference between adjacent pixels. We define $\mathcal{L}_{\text{smooth}}$ as follows:

$$\mathcal{L}_{\text{smooth}} := \frac{1}{HW} \sum_{h,w} \left(||\partial_x P_{h,w}|\,|_1 e^{-||\partial_x I_{h,w}||_2^2/\sigma} + ||\partial_y P_{h,w}|\,|_1 e^{-||\partial_y I_{h,w}||_2^2/\sigma} \right), \quad (3.8)$$

where ∂ and $||\cdot||_{\{1,2\}}$ denote the image gradients and $\{l_1, l_2\}$-norm. Note that the image gradient is calculated as $\partial_x I_{h,w} = I_{h,w} - I_{h,w-1}; 1 < w < W$. $P_{h,w} \in \mathbb{R}_+^N$ and $I_{h,w} \in \mathbb{R}^C$ are the vectors at (h, w) of P and I, respectively; σ is a scalar value, and we set it to 8 in this thesis. $\mathcal{L}_{\text{smooth}}$ is the same as proposed in [Godard et al., 2017].

Unlike the total variation that is used for the same purpose as $\mathcal{L}_{\text{smooth}}$ in many previous methods, $\mathcal{L}_{\text{smooth}}$ allows that the pixel has different values when the pixel values in the original input are different from an adjacent pixel value. Therefore, $\mathcal{L}_{\text{smooth}}$ encourages boundary-aware superpixel segmentation.

3.2.3 Image Reconstruction As an Auxiliary Task

Eq. (3.7) and (3.8) are enough to generate superpixels. In fact, by optimizing the parameters of the CNNs with reference to the sum of eq. (3.7) and (3.8), the proposed method generates plausible superpixels, as seen in the experiments. Moreover, our proposed method can generate a plausible result even if λ in eq. (3.7) is zero and the pixel-wise weights in eq. (3.8) are omitted (i.e., $\frac{1}{HW}\sum_{h,w}||\partial_x P_{h,w}||_1 + ||\partial_y P_{h,w}||_1$). Under the above condition, a trivial solution assigns the same one-hot value to all the pixels, such as $P_{h,w} = (1, 0, 0, \ldots, 0)$. But, as shown in Figure 3.5, the proposed method avoids the trivial solution.

However, the solution of eq. (3.7) does not depend on the input image, and eq. (3.8) strongly depends on the edge. Therefore, we propose reconstruction loss to make the generated image-specific superpixels. The reconstruction loss is as follows:

$$\mathcal{L}_{\text{recons}} := \frac{1}{HWC}\sum_{h,w,c}(I_{h,w,c} - \hat{I}_{h,w,c})^2. \tag{3.9}$$

This is the same as the DIP objective function.

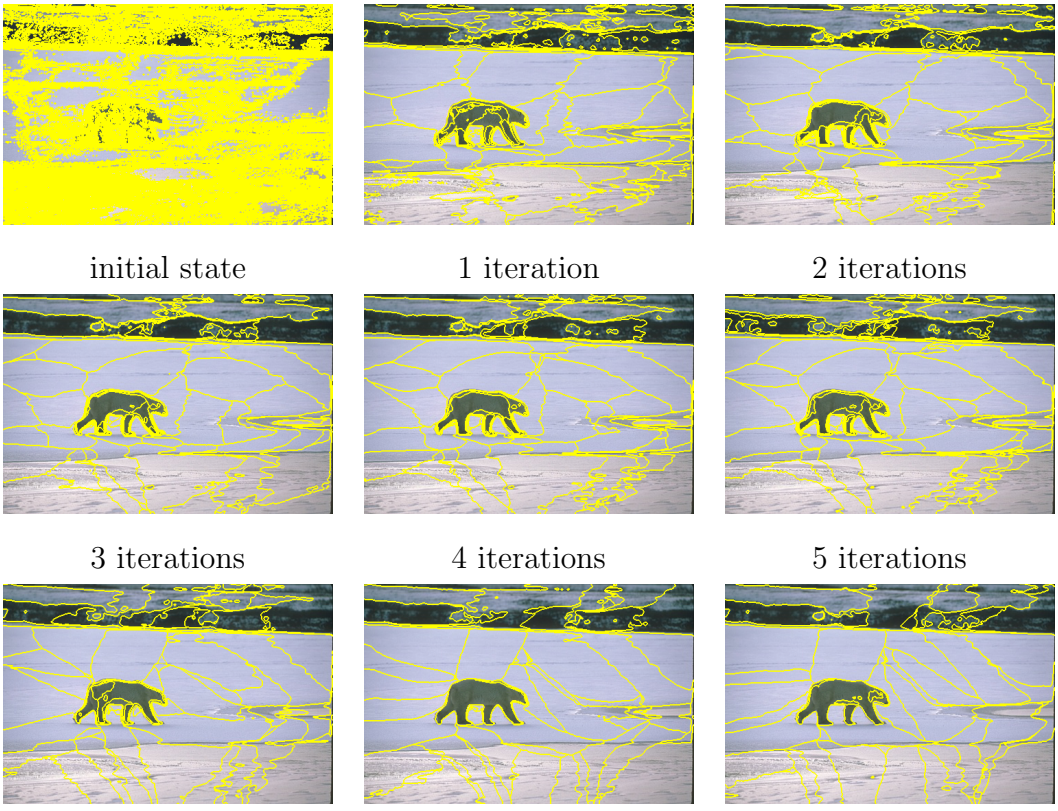
It is an irrelevant task for the superpixel segmentation to minimize the reconstruction loss. However, adding the reconstruction loss into the objective function empirically improves the superpixel quality. We evaluate its improvement in the experiments.

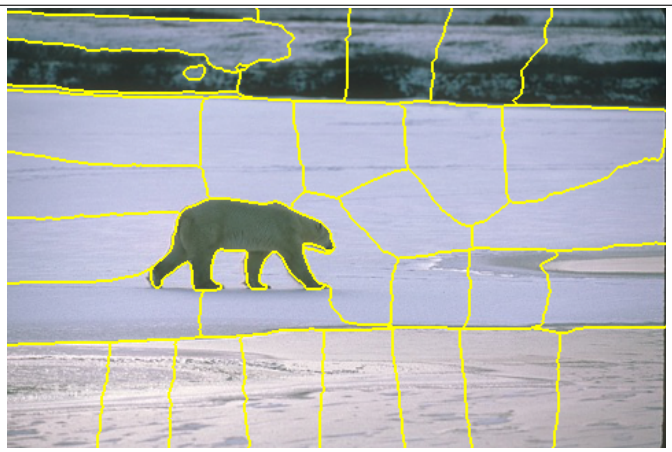
3.2.4 Empirical Studies

As the DIP uses early stopping [Morgan and Bourlard, 1990] to plausibly solve the inverse problem, our CNN-based superpixel segmentation also uses two techniques: coordinate input and instance normalization [Ulyanov et al., 2016]. Thus, we introduce the techniques and justify them in this section,

Coordinate input. We input not only the original image but also the pixel coordinate, because the convolution operation has the translation invariant, and it causes unconnected components.

Use of instance normalization. Instance normalization [Ulyanov et al., 2016] is the normalization method for CNNs. It normalizes





Obtained superpixels (100 iterations)

Figure 3.5: The result of the proposed method with $\lambda = 0$, $\alpha = 0$, and the smoothness loss without the pixel-wise weights. The proposed method avoids the trivial solution and generates plausible superpixels even if the objective function consists of only the pixel-wise entropy.

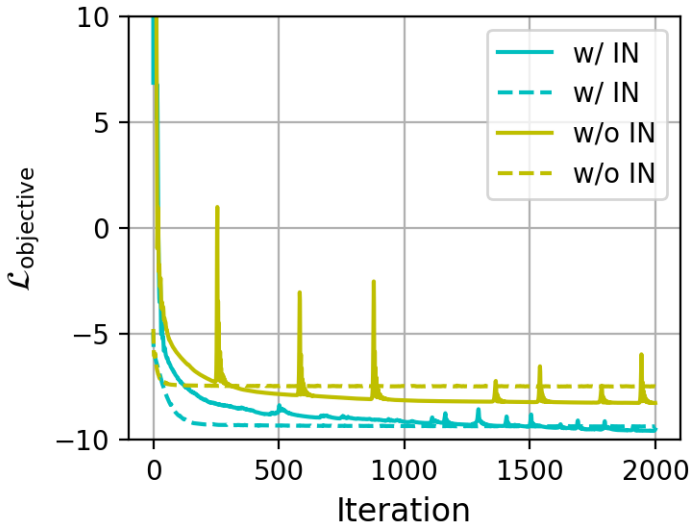


Figure 3.6: $\mathcal{L}_{\text{objective}}$ of various settings. The solid lines indicate w/ $\mathcal{L}_{\text{recons}}$, and the dashed lines indicate w/o $\mathcal{L}_{\text{recons}}$. Instance normalization (IN) layer improves $\mathcal{L}_{\text{recons}}$ in both methods.

the input whose mean and variance are 0 and 1, respectively. The detail of the normalization layer is available in Appendix B.1. Because of instance normalization, the entropy of the probability P depends on the affine parameters in the instance normalization layer. As a result, the optimization becomes slightly easy, and the model can find a better solution. We show this fact in Figure 3.6.

The pseudo-code of our algorithm is available in Algorithm 1.

3.3 Evaluation

The goal of the evaluation is to verify that the CNN can capture the non-local and high-level features. We evaluate our proposed method on the popular benchmarks [Stutz et al., 2018].

Algorithm 1 CNN-based superpixel segmentation

Input: An image $I \in \mathbb{R}^{H \times W \times C}$; pixel locations $X \in \mathbb{R}^{H \times W \times 2}$; hyperparameters: the number of superpixels N, coefficients (λ, α, β), number of iteration T, and learning rate η
Output: Superpixels $\mathcal{S}$
Initialize CNN with randomly sampled parameters θ,
$f_\theta : (I, X) \to (P, \hat{I})$
for $t = 1, \ldots, T$ **do**
    Get probability and reconstructed image,
    $(P, \hat{I}) \leftarrow f_\theta(I, X)$
    Calculate $\mathcal{L}_{\text{objective}}$
    Update parameters by gradient descent,
    $\theta \leftarrow \theta - \eta \frac{\partial}{\partial \theta} \mathcal{L}_{\text{objective}}$
end for
Assign superpixels,
$s_k = \{x_{h,w} | k = \arg\max_n P_{h,w,n}\}$

3.3.1 Metrics

We use standard metrics for superpixel segmentation to evaluate the performance, achievable segmentation accuracy (ASA), and boundary recall (BR). ASA is a metric to quantify the achievable accuracy of superpixel-based segmentation. It is defined as follows:

$$ASA(\mathcal{S}, \mathcal{G}) := \frac{\sum_i \max_j |s_i \cap g_j|}{\sum_i |g_i|}, \tag{3.10}$$

where $\mathcal{G} = \{g_1, \ldots, g_M\}$ denotes a set of ground-truth segments and g_m denotes a set of pixels. $\mathcal{S} = \{s_i\}$ denotes a set of superpixels, and s_i denotes a set of pixels; $|\cdot|$ indicates the size of the set.

BR quantifies the recall of the boundary between segments in ground-truth labels. BR is defined as follows:

$$BR(\mathcal{B}^{\mathcal{S}}, \mathcal{B}^{\mathcal{G}}) := \frac{\text{TP}(\mathcal{B}^{\mathcal{G}}, \mathcal{B}^{\mathcal{S}})}{\text{TP}(\mathcal{B}^{\mathcal{G}}, \mathcal{B}^{\mathcal{S}}) + \text{FN}(\mathcal{B}^{\mathcal{G}}, \mathcal{B}^{\mathcal{S}})}, \tag{3.11}$$

where $\mathcal{B}^{\mathcal{S}} = \{b_1^{\mathcal{S}}, \dots\}$ and $\mathcal{B}^{\mathcal{G}} = \{b_1^{\mathcal{G}} \dots\}$ denote a set of boundary pixels in $\mathcal{S}$ and $\mathcal{G}$. FN and TP are the number of false negatives and true positives boundary pixels, respectively. If a boundary pixel in $\mathcal{S}$ exists within a $(2\epsilon + 1) \times (2\epsilon + 1)$ local patch centered on an arbitrary boundary pixel in $\mathcal{G}$, the pixel is counted as TP. We set ϵ to 1 in our experiments.

3.3.2 Implementation Details

We evaluate the proposed method with the five-layer CNN with ReLU nonlinearity [Krizhevsky et al., 2012], as shown in Table 3.1. The channels for each layer except for the output layer are set to $32 \cdot 2^{l-1}$, where l indicates the layer index, $l \in [0, 5]$. We use the softmax activation to ensure $\sum_n P_{h,w,n} = 1$ and apply instance normalization [Ulyanov et al., 2016] for the feature map before the softmax activation. We optimize the model for 1,000 iterations using the Adam optimizer [Kingma and Ba, 2014]. We set 0.01 to the learning rate, and the other parameters are the same as the default parameters suggested in [Kingma and Ba, 2014]. The coefficients (λ, α, β) in eq. (3.6) are set to $(2, 2, 10)$ in our experiments, but β is zero for the model without the reconstruction cost. These parameters were roughly selected by an evaluation on the train data, and the parameters of the baseline methods were also selected in the same way.

In practice, if given only an RGB image as input, CNN groups the independent connected components as the same superpixel because CNN is the translation invariant and assigns the superpixels based on only local spatial patterns. Therefore, we also give pixel locations $X \in \mathbb{R}^{H \times W \times 2}$ to the model as input, namely, $f : (I, X) \to (P, \hat{I})$, to reduce undesired segments. The pixel locations cannot completely prevent the undesired segments, but practically, they work well. Therefore, for implementing the (RGB, location)-input (probability, reconstruction)-output model, we set input channels of CNN to 5 and output channels of CNN to $N + 3$. The inputs are normalized for each channel so that the mean and the variance of each channel of X and I are 0 and 1, respectively.

Table 3.1: The CNN architecture for superpixel segmentation. H, W, and N denote image height, image width, and the number of superpixels, respectively. The kernel size of all convolution layers are set to 3.

Layer name	Input Dimension	Output Dimension
Convolution	$H \times W \times 3$	$H \times W \times 32$
Instance Norm.	$H \times W \times 32$	$H \times W \times 32$
ReLU activation	$H \times W \times 32$	$H \times W \times 32$
Convolution	$H \times W \times 32$	$H \times W \times 64$
Instance Norm.	$H \times W \times 64$	$H \times W \times 64$
ReLU activation	$H \times W \times 64$	$H \times W \times 64$
Convolution	$H \times W \times 64$	$H \times W \times 128$
Instance Norm.	$H \times W \times 128$	$H \times W \times 128$
ReLU activation	$H \times W \times 128$	$H \times W \times 128$
Convolution	$H \times W \times 128$	$H \times W \times 256$
Instance Norm.	$H \times W \times 256$	$H \times W \times 256$
ReLU activation	$H \times W \times 256$	$H \times W \times 256$
Convolution	$H \times W \times 256$	$H \times W \times (N + 3)$
Instance Norm.	$H \times W \times N$	$H \times W \times N$
Softmax	$H \times W \times N$	$H \times W \times N$

3.3.3 Baseline Methods

We compare the proposed method to the clustering-based [Achanta et al., 2012], the energy optimization [Van den Bergh et al., 2012, Yao et al., 2015], and the graph-based methods [Felzenszwalb and Huttenlocher, 2004] on the Berkeley Segmentation Dataset and Benchmarks (BSDS500) [Arbelaez et al., 2010] and the Stanford Background Dataset (SBD) [Gould et al., 2009]. The BSDS500 contains 300 train/validation images and 200 test images, and we use 200 test images for the evaluation in the experiments, and SBD contains 715 images, and we use all images for the evaluation. We use implementations of OpenCV [Bradski, 2000], scikit-image [van der Walt et al., 2014], the author's implementation [1] of ETPS for baseline methods, and PyTorch [Paszke et al., 2019] to implement the proposed method.

3.3.4 Results

3.3.4.1 Adaptive Controlling of the Number of Superpixels

We show the number of superpixels per image with various λ values in Figure 3.7. We set N to 500. The model assigns various numbers of superpixels in each image, and the number of superpixels spreads over a wide range when λ is large. This indicates that our CNN-based method adapts the number of superpixels depending on the given images. If one desires the superpixels to be as few as possible, λ should be set to a small value. However, if one desires to adaptively control the number of superpixels, λ should be set as a large value.

3.3.4.2 Comparison with Baseline Methods

Figure 3.8 shows that the proposed method achieves comparable or better results compared to other methods in ASA with ≤ 200 superpixels. As we expected, the proposed methods clearly improve ASA with a small number of superpixels. The results indicate that the proposed method utilizes the global structure to generate superpixels. We believe that the CNNs' prior induces the properties. The

[1] https://bitbucket.org/mboben/spixel/src/master/

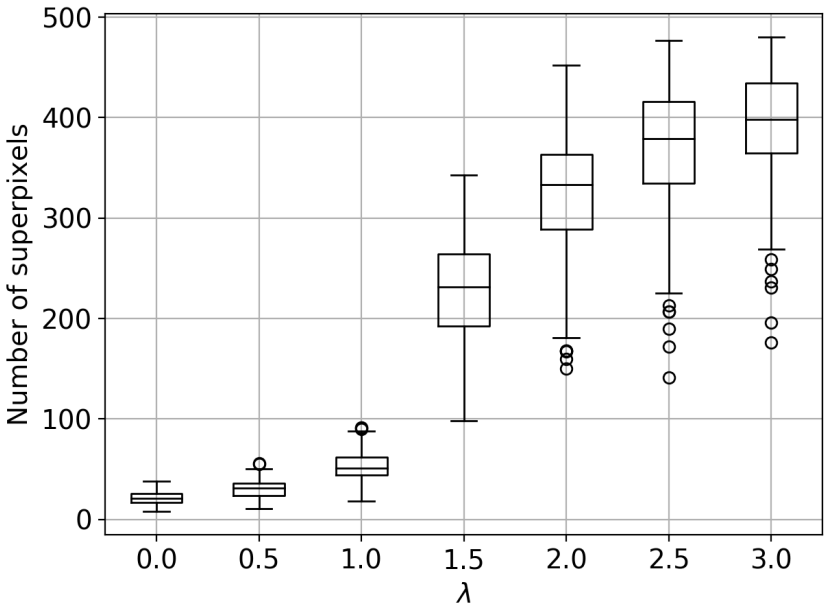


Figure 3.7: The number of superpixels per image on the BSDS500 [Arbelaez et al., 2010] test image set with various λ values. The maximum number of superpixels N is 500. The number of superpixels converges on a small number when λ is small. However, when λ is large, the number of superpixels spreads over a wide range, and the mean number of superpixels becomes large.

proposed methods also achieves comparable or better BR, with the small number of superpixels. Significantly, the proposed method with the reconstruction dramatically improves BR with ≤ 200 superpixels on the BSDS500. It indicates that the reconstruction cost refines the segmentation accuracy around the object boundaries.

We show example results of each method in Figure 3.9. The superpixels generated by the proposed methods have a similar flexibility to the FH method [Felzenszwalb and Huttenlocher, 2004] and compactness similar to SLIC [Achanta et al., 2012], especially with the reconstruction cost. The proposed method with the reconstruction partially fits the superpixels to the detail components in the images, such as the vertical tail of the helicopter and the body paint of the plane. The FH method also seems to fit segments to detail components; however, its ASA is lower than other methods, and the fact indicates that the FH method groups the pixels belonging to different segments in the ground-truth label. It indicates that the segments generated by the FH method cannot preserve the semantic information of the input image. Surprisingly, the proposed method without the reconstruction segments the object components, as shown in Figure 3.9 (e). The method does not have any handcrafted priors except for the local smoothness; hence, we believe that the CNNs' prior effectively works for the superpixel segmentation.

3.4 Conclusion

In this section, we evaluated the proposed methods and verified the CNNs' prior for superpixel segmentation. Our proposed method, the mutual information maximization framework, can capture the object components without any priors except for the local smoothness. The evidence implies that the CNN has a good prior for superpixel segmentation. Moreover, combined with the image reconstruction task, the method improves ASA and BR. As a result, our proposed method outperforms the existing methods with a small number of superpixels.

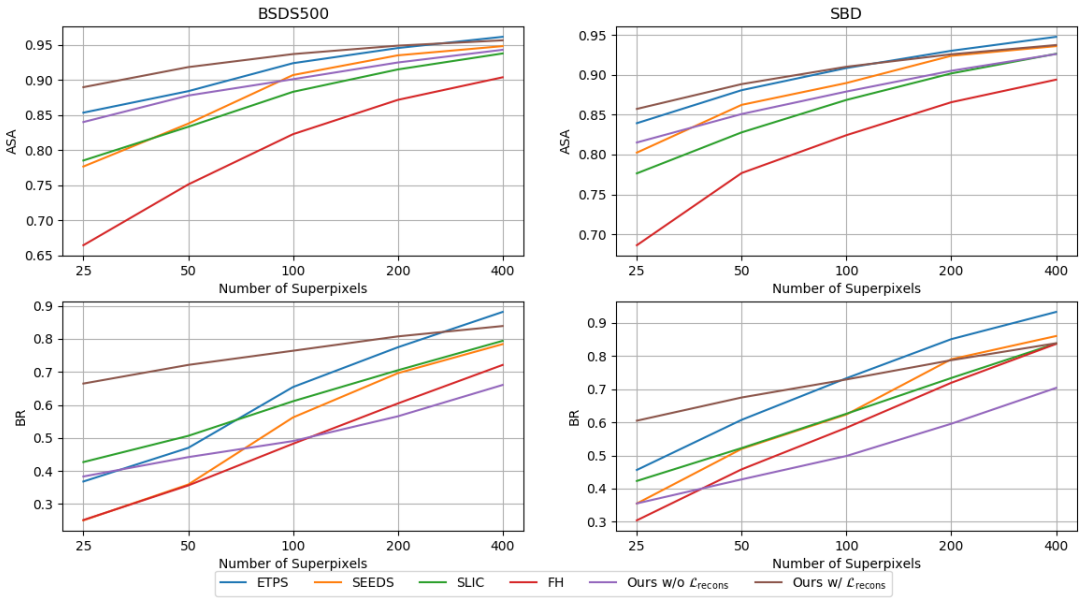
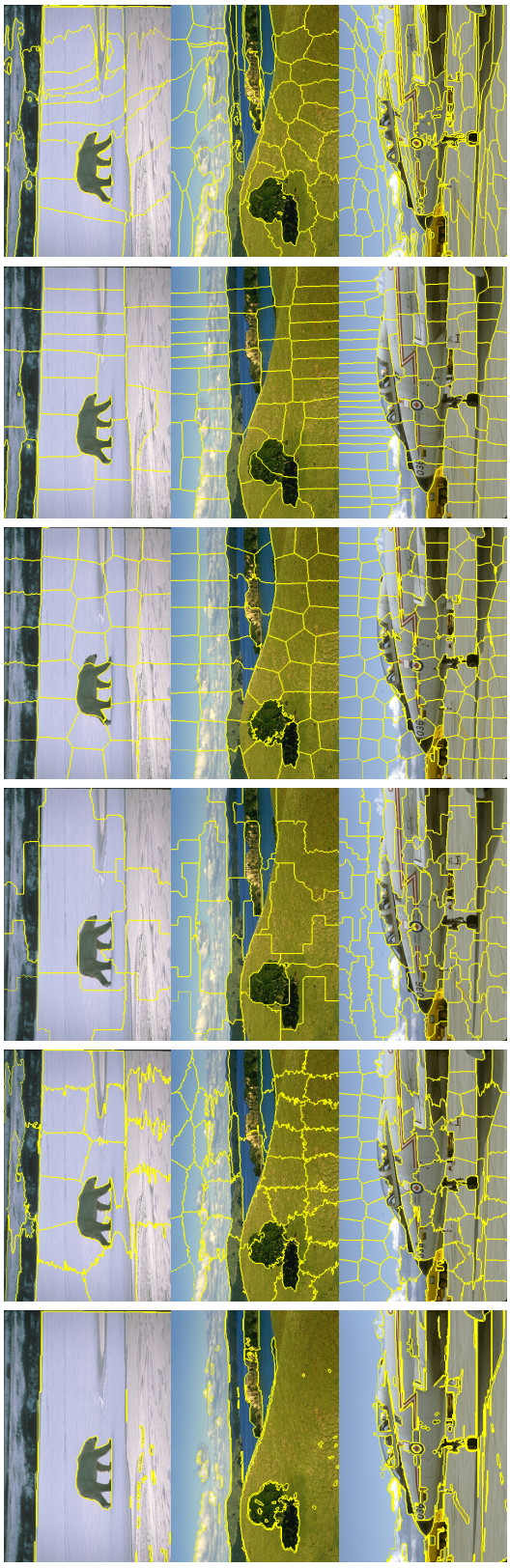


Figure 3.8: Comparison of proposed method to baseline methods [Achanta et al., 2012, Van den Bergh et al., 2012, Felzenszwalb and Huttenlocher, 2004, Yao et al., 2015]. We show achievable segmentation accuracy (ASA) and boundary recall (BR) on the BSDS500 [Arbelaez et al., 2010] and SBD [Gould et al., 2009] with various numbers of superpixels. The proposed method w/ recons and w/o recons determines whether the reconstruction cost is used for optimization or not.



(a) (b) (c) (d) (e) (f)

Figure 3.9: Example results of the (a) FH [Felzenszwalb and Huttenlocher, 2004], (b) SLIC [Achanta et al., 2012], (c) SEEDS [Van den Bergh et al., 2012], (d) ETPS [Yao et al., 2015], (e) ours without the reconstruction const, and (f) ours with the reconstruction cost. From top to bottom, the number of superpixels is 25, 50, and 100.

4

Superpixel-based Image Segmentation by Superpixel Convolution

Image segmentation is the fundamental task in the image processing and computer vision. To capture high-context and global information, many recent proposed methods leverage CNNs [Long et al., 2015, Zhao et al., 2017, Chen et al., 2017a, Chen et al., 2014, Yu and Koltun, 2015]. Although CNNs are useful in capturing data, it discards detailed information, such as high frequency components, object boundaries, and small and/or thin objects, due to the downsampling operation. Therefore, generated segments are roughly fit to the object and backgrounds. Several techniques have been proposed to preserve information [Yu and Koltun, 2015, Chen et al., 2014, Li et al., 2020], but they require extra computational costs or complex procedures. Empirically, dilated convolution, also known as atrous convolution, can be used to eliminate downsampling operations from a network, but it causes the computational costs to explode.

Classically, as described in the previous section, superpixels are used to reduce computational costs. They are a low-dimensional representation of images and generally given as a set of pixels similar in color and other low-level properties. Our strategy is to leverage superpixels instead of general downsampling methods,

such as pooling and convolution with a stride.

[Kwak et al., 2017] proposed the superpixel-based downsampling method, or superpixel pooling, and they use it for weakly supervised semantic segmentation and improve the segmentation quality. However, they deal with downsampled images as the vectors (not as images), because there is no means to process the superpixel image by the convolution operation.

Therefore, we propose *superpixel convolution*, which is a convolution operation designed for images downsampled with superpixels. Because of superpixel covolution, we can build CNNs replaced fixed-kernel downsampling operations with a superpixel-based downsampling operator.

In the rest of this chapter, we first introduce graph convolution, which is closely related to our proposed superpixel convolution, and we describe the differences between them. Then, we propose the superpixel convolution and dilated superpixel convolution and evaluate them.

4.1 Convolutional Neural Networks on Graphs

4.1.1 Adjacency Matrix and Graph Laplacian

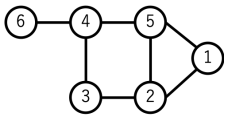
Let $\mathcal{G} = (\mathcal{E}, \mathcal{V})$ be an undirected, weighted graph that consists of a set of edges $\mathcal{E}$ and a set of vertexes $\mathcal{V}$. Each edge $e_{ij} \in \mathcal{E}$ has a weight $w_{ij} \in \mathbb{R}$, and each vertex $v_i \in \mathcal{V}$ has a feature vector $f_i \in \mathbb{R}^N$. Let the number of vertexes $|\mathcal{V}|$ be M, and then we define an adjacency matrix $W \in \mathbb{R}^{M \times M}$ as follows:

$$W := \begin{cases} w_{ij} & \text{if nodes } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}, \tag{4.1}$$

and a degree matrix is also defined as $D_{ii} = \sum_j W_{ij}; D \in \mathbb{R}^{M \times M}$. Note that the adjacency matrix is a non-negative symmetric matrix (i.e., $w_{ij} = w_{ji} \geq 0$), and the degree matrix is a diagonal matrix. Then, the unnormalized graph Laplacian L is defined as follows:

$$L := D - W. \tag{4.2}$$

We illustrate each matrix in Figure 4.1.



$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

(a) (b) (c) (d)

Figure 4.1: Illustration of the unnormalized graph Laplacian. From left to right, (a) the graph structure, (b) the degree matrix, (c) the adjacency matrix, and (d) the unnormalized graph Laplacian.

Then, we also define the normalized graph Laplacian L_{norm} and the random work graph Laplacian L_{rw} as follows:

$$L_{\text{norm}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \tag{4.3}$$

$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W \tag{4.4}$$

We provide some propositions for the normalized and unnormalized graph Laplacian in the appendix.

4.1.2 Superpixel Images As Graphs

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_K\}$ be a set of superpixels. We regard superpixels as the graph. A set of vertexes corresponds to a set of superpixels. We introduce the edge $w_{ij} = 1$ between superpixels s_i and s_j, if s_i and s_j share the their boundaries or $i = j$. Therefore, the graph of superpixel images is defined as $\mathcal{G} = (\mathcal{E}, \mathcal{S})$, and its adjacency matrix is as follows:

$$W_{ij} := \begin{cases} 1 & \text{if superpixels } i \text{ and } j \text{ share their boundaries} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The node feature, which means the feature vector of superpixels $\{s_k\}$, is made by superpixel pooling, which is described in the next section.

4.1.3 Superpixel Pooling

Superpixel pooling was originally introduced by [Kwak et al., 2017] is a down-sampling method that receives an image or an intermediate representation of CNNs and precomputed superpixels as input and returns the feature vectors on the superpixels. Therefore, we define the superpixel pooling as a map, $f : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{N\times C}$, where H, W, and C denote the input image height, the width, and the channels, and N denotes the number of superpixels.

The general pooling operations sample the specific value within the predefined fixed area. In the neural network scenario, max pooling and average pooling, which are defined as follows, are basically used:

$$y_i := \max_{j\in\mathcal{N}(i)} x_j, \tag{4.6}$$

$$y_i := \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} x_j, \tag{4.7}$$

where $\mathcal{N}(i)$ denotes a set of neighbor pixels of the i-th pixel, and y_i denotes the feature vector on the i-th pixel of the output, and x_j also denotes the feature vector on the j-th pixel of the input. In other words, the max and average pooling sample max value and average value over the neighbor pixel features.

We can easily extend the pooling into the superpixel image domain. We define the feature vector on the i-th pixel as x_i and the feature vector on a i-th superpixel as f_i. Then, we define superpixel max and average pooling as follows:

$$f_i := \max_{j\in s_i} x_j, \tag{4.8}$$

$$f_i := \frac{1}{|s_i|}\sum_{j\in s_i} x_j. \tag{4.9}$$

Our strategy is to describe superpixel images as graphs and to process them with a convolution operator on graphs. Therefore, in the next section, we first review graph convolutions, which are defined as convolutions on graphs, and next, we propose *superpixel convolution*, which is a graph convolution for superpixel images.

4.1.4 Graph Convolution

In this section, we introduce graph convolutions which have recently studied in the machine learning field. Graph convolutions are roughly classified into a spectral domain and a spatial domain, and in the following section, we introduce graph convolutions for each domain.

4.1.4.1 Spectral Domain

To define graph convolution in the spectral domain, we first introduce graph Fourier transformation. The following derivation is based on [Bruna et al., 2013], but the graph Fourier transformation was originally introduced by [Hammond et al., 2011].

In the graph signal domain, a natural definition of the local smoothness at the i-th node is as follows:

$$||\nabla f_i||_W^2 := \sum_j W_{ij}(f_i - f_j)^2, \tag{4.10}$$

where $F = (f_1, \ldots, f_i, \ldots, f_M)^\top \in \mathbb{R}^{M \times C}$ denotes the feature vector on the i-th node. This definition corresponds to the quadratic form of the graph Laplacian L. Thus, the eigenvector of L corresponds to

$$v_i = \underset{f_i \in \mathbb{R}^N \ ||f_i||=1 \ x \perp \{v_a, \ldots, v_{i-1}}{\arg\min} ||\nabla f_i||_W^2, \tag{4.11}$$

and the corresponding eigenvalues λ_i allows the smoothness of a vector f_i to be read off from the coefficients of f_i in $\{v_1, \ldots, v_M\}$. Note that M corresponds to the number of vertexes. As a result, the eigenvalues are regarded as the Fourier coefficients of a graph following an adjacency matrix W, and the eigenvectors are also regarded as the Fourier bases. Therefore, the graph Fourier transformations of the feature vectors on the vertexes are given as follows:

$$\mathcal{F}(F, L) := U^\top F, \tag{4.12}$$

where $U^\top$ denotes the eigenvectors of the graph Laplacian L, $U = (u_1, \ldots, u_M); U \in \mathbb{R}^{M \times M}$. Since the graph Laplacian is the symmetric semi-positive definite matrix, U is the orthogonal matrix, and then the inverse trans-

formation of the graph Fourier transformation is defined as $\mathcal{F}^{-1}(\hat{F}, L) := U\hat{F} = UU^\top F = F$.

Next, we introduce the spectral domain graph convolution introduced by [Bruna et al., 2013]. The graph convolution in the spectral domain is defined as follows:

$$\hat{F}_{:,c'} := U \sum_{c=1}^{C} \Lambda_{c',c} U^\top F_{:,c}, \tag{4.13}$$

where $\hat{F} \in \mathbb{R}^{M \times C'}$ denotes the output feature vectors, and C' is the number of output dimension; $\Lambda_{c',c} \in \mathbb{R}^{M \times M}$ denotes the convolution filter, which is the diagonal matrix. Note that $F_{:,c}$ denotes the c-th column of F. Classically, the filter parameters are determined by solving an optimization problem (e.g.[Shuman et al., 2013]) However, [Bruna et al., 2013] proposed a graph convolution in a machine learning scenario. Therefore, the filter parameter Λ is trained through a gradient descent with backpropagation.

The graph convolution proposed by [Bruna et al., 2013] requires high computational costs because an eigen decomposition whose computational costs is $\mathcal{O}(n^2)$ is needed. Therefore, [Defferrard et al., 2016] proposed a more efficient formulation with Chebyshev expansion. The graph convolution approximated Chebyshev expansion is traditionally used in graph signal processing [Hammond et al., 2011], and Defferrard et al. modified it for graph neural networks.

The graph convolution proposed by Defferrard et al. approximates eq. (4.13) as follows:

$$\hat{F}_{:,c'} = U \sum_{c}^{C} \Lambda_{c',c} U^\top F_{:,c} \approx \sum_{c}^{C} \sum_{k=1}^{K} \theta_{c,c',k} T_k(\tilde{L}) F_{:,c}, \tag{4.14}$$

where $\theta \in \mathbb{R}^{C \times C' \times K}$ is the trainable parameter, and $\tilde{L} = \frac{2L}{\lambda_{\max}} - I$. Note that $\lambda_{\max}$ is the largest eigenvalue of L. $T_k(\cdot)$ denotes Chebyshev polynomials with the k-th order, which are defined as follows:

$$\begin{aligned} T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x), \\ \text{with } T_0(x) &= 1 \text{ and } T_1(x) = x. \end{aligned} \tag{4.15}$$

This formulation has three advantages: (i) trainable parameters decrease from $C'CN$ to $C'CK$, (ii) computational costs also decrease since it does not require eigen decomposition, and (iii) it is designed as localized convolutional filters, meaning that the Chebyshev polynomial of order K corresponds to the filter size of general spatial convolutions.

Empirically, eq. (4.14) is well approximated with $K = 2$. Moreover, [Kipf and Welling, 2016] proposed a simpler formulation of eq. (4.14) by representing the two trainable parameters as a shared parameter and introducing a self-loop into the node. When $K = 2$, eq. (4.14) is represented as follows:

$$\sum_{c}^{C}\sum_{k=1}^{2}\theta_{c,c',k}T_k(\tilde{L})F_{:,c} = (F\theta_0)_{:,c'} - D^{-1/2}AD^{-1/2}(F\theta_1)_{:,c'}, \tag{4.16}$$

and Kipf and Welling approximated the trainable parameters $\theta_0 \in \mathbb{R}^{C\times C'}$ and $\theta_1 \in \mathbb{R}^{C\times C'}$ as $\theta = \theta_0 = -\theta_1$. Note that to derive eq. (4.16), Kipf and Welling approximated $\lambda_{\max} \approx 2$. Then, eq. (4.16) is approximated as follows:

$$F\theta_0 - D^{-1/2}AD^{-1/2}F\theta_1 \approx (I - D^{-1/2}AD^{-1/2})F\theta. \tag{4.17}$$

Moreover, they also proposed a *reparameterization trick* to reduce the risk of a vanishing gradient: $I + D^{-1/2}AD^{-1/2} \rightarrow \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, with $\tilde{A} = I + A$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. This reparameterization indicates the self-loop for each node. As a result, Kipf and Welling's formula for a graph convolution is as follows:

$$\hat{F} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}F\theta. \tag{4.18}$$

4.1.4.2 Spatial Domain

[Bruna et al., 2013] also proposed graph convolutions in a spatial domain. We first review the convolution for image data and then introduce the spatial domain graph convolution introduced by [Bruna et al., 2013].

A spatial convolution in an image domain is defined as follows:

$$\hat{x}_{h,w,c'} := \sum_{c}^{C}\sum_{\delta y,\delta x=-\lfloor k/2\rfloor}^{\lfloor k/2\rfloor} w_{\delta y,\delta x,c',c}x_{h+\delta y,w+\delta x,c}, \tag{4.19}$$

where w denotes filter parameters, and k denotes the filter size; $\lfloor \cdot \rfloor$ is a floor function. Note that, to be exact, eq. (4.19) is called a "*cross correlation*," but in the machine learning community, eq. (4.19) is called a "*convolution*." Therefore, we also call eq. (4.19) a convolution in this thesis.

We regard a spatial convolution as an aggregation of the neighbor pixel features and extend it to a graph convolution in the spatial domain. Let $\mathcal{N}(i)$ be the neighbor pixels of the i-th pixel and x_j be the feature vectors of the j-th pixel. Then, a graph convolution in a spatial domain is defined as follows:

$$\hat{x}_i = \sum_{j \in \mathcal{N}(i)} W_{ij} x_j, \tag{4.20}$$

where W denotes the adjacency matrix, but [Bruna et al., 2013] regard it as trainable parameters. Note that although W is the trainable parameters, the graph structure is known, meaning that W_{ij} is set to 0 if no edge exists between the i-th node and the j-th node.

4.1.4.3 Convolutions in Image Domains as Graph Convolutions

When we consider pixels as graph vertexes and introduce the edges between neighbor pixels, the image is also regarded as a graph. Therefore, if we design the adjacency matrix with local connectivity between the pixels, we can describe a spatial convolution using the adjacency matrix as follows:

$$\hat{X}_{c'} = \sum_{c}^{C} W_{c',c} X_c, \tag{4.21}$$

where $X \in \mathbb{R}^{C \times M}$ denotes the feature vectors on the pixels, and $\{W_{c',c} \in \mathbb{R}^{M \times M}\}_{c',c}$ denotes the trainable adjacency matrix. Note that the adjacency matrix includes a self-loop.

Unlike a graph convolution in a spatial domain, W_{ij} is the matrix similar to the circulant matrix. The circulant matrix is defined as follows:

$$C := \begin{pmatrix} c_1 & c_2 & \cdots & c_{M-1} & c_M \\ c_M & c_1 & \cdots & c_{M-2} & c_{M-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_3 & c_4 & \cdots & c_1 & c_2 \\ c_2 & c_3 & \cdots & c_M & c_1 \end{pmatrix}. \tag{4.22}$$

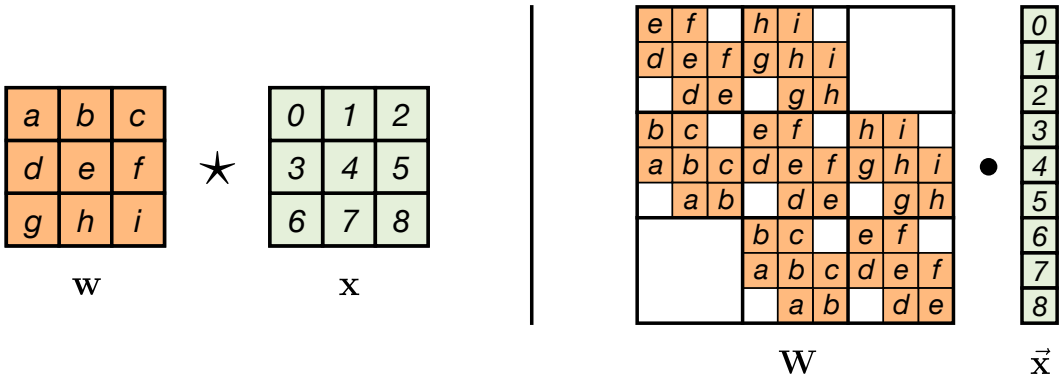


Figure 4.2: Illustration of the unrolled convolution defined in eq. (4.21). The left figure denotes a convolution between a filter $\mathbf{w}$ and an input $\mathbf{x}$. The right figure denotes an unrolled convolution. The figure is taken from [Hoogeboom et al., 2019].

Therefore, a spatial convolution with a circulant matrix has only M parameters, $\{c_1, c_2, \ldots, c_M\}$. We illustrate the unrolled convolution defined in eq. (4.21) in Figure 4.2.

Moreover, because of the sparsity of the adjacency matrix, the trainable parameters per channel are less than M, which is much less than the graph convolution in the spatial domain. We believe that superpixel images also have these properties. Thus, we leverages the properties to efficiently process superpixel images with a graph convolution.

4.2 Superpixel Convolution

We propose generalized frameworks of convolution and dilate convolution [Yu and Koltun, 2015] in the image domain, as shown in Figure 4.3.

Let $W \in \{0,1\}^{M \times M}$ be the adjacency matrix defined as eq. (4.5) and $\{C_{c',c} \in \mathbb{R}^{M \times M}\}_{c',c}$ be the circulant matrix whose parameters are trainable. We define the

graph convolution for superpixel images as follows:

$$\hat{F}_{:,c'} := \sum_{c} \left(C_{c',c} \circ W^K \right) F_{:,c}, \tag{4.23}$$

where $\circ$ denotes the Hadamard or element-wise product. Note that we compute W^K in Boolean algebra; that is the K-th power of W is calculated with *logical or* and *logical and*. Then, K corresponds to the kernel size because the elements of the K-th power of the adjacency matrix indicate the number of paths from the i-th node to the j-th node with a path length of K. Moreover, since W has a self-loop, if the path length between the i-th node and the j-th node is less than K, $W \neq 0$. Because we compute W^K in Boolean algebra, W represents whether the path length is less than K. We call eq. (4.23) *superpixel convolution*. Note that if F is the image, the superpixel convolution corresponds to the spatial convolution.

The trainable parameters of the superpixel convolution are $\{C_{c,c'} \in \mathbb{R}^{M \times M}\}$, and each of them has only M parameters since we define it as the circulant matrix. It is reasonable to define the trainable parameter as the circulant matrix because the superpixel images are more spatially structured compared to general graphs. Therefore, the superpixel convolution is more similar to the spatial convolution than the graph convolution in a spatial domain.

The trainable parameter $C_{c',c}$ can be trained by the gradient method through backpropagation. Let L be the predefined objective function, and then the gradients of parameters are given as follows:

$$\frac{\partial L}{\partial C_{c',c,i,j}} = W^K_{i,j} F_{j,c} \frac{\partial L}{\partial \hat{F}_{i,c'}}, \tag{4.24}$$

$$\frac{\partial L}{\partial F_{j,c}} = \sum_{c'} \left(C_{c',c,i,j} \circ W^K_{i,j} \right) \frac{\partial L}{\partial \hat{F}_{i,c'}}, \tag{4.25}$$

where $W^K_{i,j}$ denotes the (i,j) element of the matrix W^K.

4.2.1 Dilated Superpixel Convolution

The dilated convolution proposed by [Yu and Koltun, 2015] is widely used for several image recognition tasks, such as image segmentation [Zhao et al., 2017,

Chen et al., 2017b] and aims to effectively expand the receptive field. The dilated convolution is defined as follows:

$$\hat{x}_c := \sum_{c}^{C} \sum_{\delta y, \delta x = -\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} w_{\delta y, \delta x, c', c} x_{h + d\delta y, w + d\delta x, c}, \tag{4.26}$$

where $d \in \mathbb{N}$ denotes a dilation parameter. When $d = 1$, the dilated convolution corresponds to the standard spatial convolution.

We propose the dilated convolution for superpixel images, or *dilated superpixel convolution*, as aggregation of K-hop nodes. The dilated superpixel convolution is defined as follows:

$$\hat{F}^{dilate}_{:,c'} := \sum_{c} \left(C_{c',c} \circ \hat{W}^{(K)} \right) F_{:,c}, \tag{4.27}$$

$$\hat{W}^{(K)} := I + W^K - W^{K-1}, \tag{4.28}$$

where $\hat{W}$ has 1, if the path length from the i-th node to the j-th node is K or $i = j$, otherwise 0. The aim of the dilated superpixel convolution is also to expand the receptive field with sparse connections.

The trainable parameters can also be trained by the gradient descent, and their gradients are given as follows:

$$\frac{\partial L}{\partial C_{c',c,i,j}} = \hat{W}^{(K)}_{i,j} F_{j,c} \frac{\partial L}{\partial \hat{F}_{i,c'}}, \tag{4.29}$$

$$\frac{\partial L}{\partial F_{j,c}} = \sum_{c'} \left(C_{c',c,i,j} \circ \hat{W}^{(K)}_{i,j} \right) \frac{\partial L}{\partial \hat{F}^{dilate}_{i,c'}}. \tag{4.30}$$

4.2.2 Weight Decomposition for Efficient Parameterization

The kernel size of convolution layers in general CNNs is generally 3×3, and the number of their parameters is $9CC'$. The number of superpixels is usually between 100 and 10,000, and the number of parameters in the superpixel convolution is up to $10000CC'$. Therefore, the superpixel convolution may be infeasible when the number of superpixels is large.

We propose a decomposed superpixel convolution to prevent an explosion in the number of parameters. We parameterize the weight matrix $C_{c',c}$ with the bases $B \in \mathbb{R}^{P \times M \times M}$ and the coefficient $T \in \mathbb{R}^{C \times C' \times P}$ as follows:

$$C_{c',c} = \sum_{p}^{P} T_{c',c,p} B_p. \tag{4.31}$$

Only the bases depend on the number of superpixels, and if the number of bases P is small, the number of trainable parameters is tractable. Note that B is also the circulant matrix; hence, the number of trainable parameters is only $P \times M$.

4.3 Evaluation

The goal of the evaluation is to verify that superpixels potentially improve the CNNs' performance for segmentation tasks. Thus, we compare the general CNN and the CNN using our proposed superpixel convolution instead of the general convolution operations. We use HKU-IS [Li and Yu, 2015], which is a dataset for two class segmentation. It contains 2,500 training data, 500 validation data, and 1447 test data.

4.3.1 Metrics

We use BR as defined in (3.11), mean absolute error (MAE), and F-scores as the evaluation metrics. MAE and F-scores are defined as:

$$MAE := \frac{1}{H \times W} \sum_{h} \sum_{w} |p_{h,w} - g_{h,w}|, \tag{4.32}$$

$$F_\beta := \frac{(1 + \beta^2) Precision \times Recall}{\beta^2 Precision + Recall}, \tag{4.33}$$

where $p \in \{0,1\}^{H \times W}$ and $g \in \{0,1\}^{H \times W}$ denote the prediction and the ground truth, respectively. *Precision* and *Recall* are defined as:

$$Precision = \frac{TP}{TP + FP}, \tag{4.34}$$

$$Recall = \frac{TP}{TP + FN}, \tag{4.35}$$

where TP, FP, and FN denote the number of true positive, false positive, and false negative pixels. As suggested by previous works, we set β^2 to 0.3 to emphasize the importance of the precision value.

4.3.2 Implementation Details

To show the effectiveness of our superpixel convolution, we compare it to a general CNN. All the models in the experiment have the same architecture (Table 4.1). Superpixels are calculated by SLIC [Achanta et al., 2012]. We optimize models using the Adam optimizer [Kingma and Ba, 2014] with the following binary cross entropy loss for 30 epochs.

$$\mathcal{L}_{BCE}(m, t) := \frac{1}{H \times W} \sum_h \sum_w (g_{h,w} \log(m_{h,w}) + (1 - g_{h,w}) \log(1 - m_{h,w})), \tag{4.36}$$

where $m \in (0, 1)^{H \times W}$ denotes the raw output of the model. We set the hyperparameters for the Adam optimizer to the default parameters suggested in [Kingma and Ba, 2014]. We evaluate models every epoch on validation data and use the best model for comparison.

4.3.3 Results

We compare the superpixel convolution (referred as SC or SConv) and the dilated superpixel convolution (referred as DSC). The DSC model has the same parameter as the model in Table 4.1, and the SC model replaces all dilated superpixel convolutions with a superpixel convolution with a kernel size of $k = 1$. We show the results in Table 4.2. Note that no decomp indicates the superpixel convolution without the weight decomposition defined in eq. (4.31).

In SC, the scores improve in proportion to the increase in parameters p. Interestingly, DSC has an inverse property. We assume that this may be due to the effect of regularization by p. Note that a large number of parameters seem to be unnecessary when aggregating a wide range of pixels.

Table 4.1: Model architectures for the foreground segmentation. The parameters in parentheses indicate baseline CNN parameters. SConv indicates a superpixel convolution, and DSConv indicates a dilated superpixel convolution. Kernel size indicates height × width of the convolution kernels, and output size indicates channels × height × width of the output map. All the blocks have shortcut connections, the same as ResNet [He et al., 2016a, He et al., 2016b].

Modules	Kernel size	Output size	Modules	Kernel size	Output size
Conv1_{1,2}			Block3_{1,2,3}		
Conv	3×3	64×256×256	Conv	1×1	32×256 (32×16×16)
ReLU		64×256×256	Tanh		32×256 (32×16×16)
Conv2			DSConv	4 (3×3, dilate 4)	32×256 (32×16×16)
Conv	3×3	32×256×256	Tanh		32×256 (32×16×16)
ReLU		32×256×256	Conv	1×1	128×256 (128×16×16)
Max Pooling	256 region (16×16)	32×256 (32×16×16)	Tanh		128×256 (128×16×16)
Block1_{1,2,3}			Conv3		
Conv	1×1	8×256 (8×16×16)	Conv	1×1	32×256 (32×16×16)
Tanh		8×256 (8×16×16)	ReLU		32×256 (32×16×16)
SConv	1 (3×3)	8×256 (8×16×16)	Up-Sampling		32×256×256
Tanh		8×256 (8×16×16)	Concatenation	(with Conv2 output)	64×256×256
Conv	1×1	32×256 (32×16×16)	Conv4		
Tanh		32×256 (32×16×16)	Conv	1×1	64×256×256
Block2_{1,2,3}			ReLU		64×256×256
Conv	1×1	16×256 (16×16×16)	Conv5		
Tanh		16×256 (16×16×16)	Conv	1×1	1×256×256
DSConv	2 (3×3, dilate 2)	16×256 (16×16×16)	Sigmoid		1×256×256
Tanh		16×256 (16×16×16)			
Conv	1×1	64×256 (64×16×16)			
Tanh		64×256 (64×16×16)			

Table 4.2: Comparison between SC and DSC.

Model	P	BR	MAE	F_β-Score
SC no decomp	-	0.255	0.143	0.744
SC	3	0.266	0.141	0.728
	9	0.290	0.140	0.730
	18	0.279	0.140	0.733
DSC no decomp	-	0.322	0.118	0.794
DSC	3	0.362	0.118	0.797
	9	0.362	0.121	0.784
	18	0.356	0.123	0.783

Table 4.3: Comparison between SC and CNNs.

Model	BR	MAE	F_β-Score
CNN	0.201	0.140	0.719
Dilated CNN	0.270	0.125	0.775
SC ($p = 9$)	0.290	0.140	0.730
DSC ($p = 9$)	0.362	0.121	0.784

4.3.4 Comparison of SC and CNNs

We compare our SC model and baseline CNNs. We use two models as baseline models; one is a model defined in Table 4.1, and the other is a model replacing the convolution layers with dilated convolution layers. We show the results and the precision-recall curve in Table 4.3 and Figure 4.4, respectively. The results show that both our SC and DSC models outperform the baseline models. In particular, our SC and DSC achieve better BR than the baseline models. This fact indicates that our model can preserve detailed information, namely, object boundaries.

Finally, we show example results in Figure 4.5. The SC and DSC improve the accuracy of object boundaries and the small objects compared to the baseline

methods. However, the performance of the proposed method depends on the superpixel quality. In this experiment, we utilize SLIC to generate the superpixels utilizing color similarity. Therefore, it is difficult to distinguish between the foreground and background which have the same color, as shown in Figure 4.6. Thus, better superpixel algorithms or end-to-end frameworks are needed.

4.4 Conclusion

In this section, we evaluated the superpixel convolution and dilated superpixel convolution on a foreground segmentation task. The proposed method outperforms baseline CNNs, especially in terms of the BR score, because the superpixels preserve the boundary information. The fact implies the use of superpixels may improve CNNs' performance for segmentation tasks.

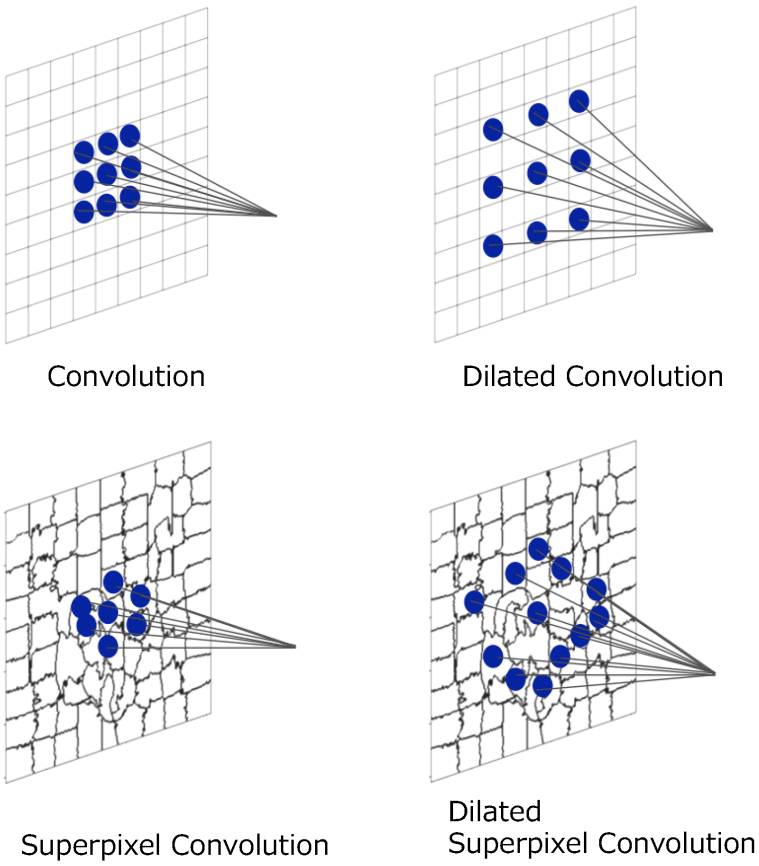


Figure 4.3: Illustration of convolution, dilated convolution, our proposed superpixel convolution, and dilated superpixel convolution. Because the arrangement of superpixels is irregular, the aggregated elements depend on the position.

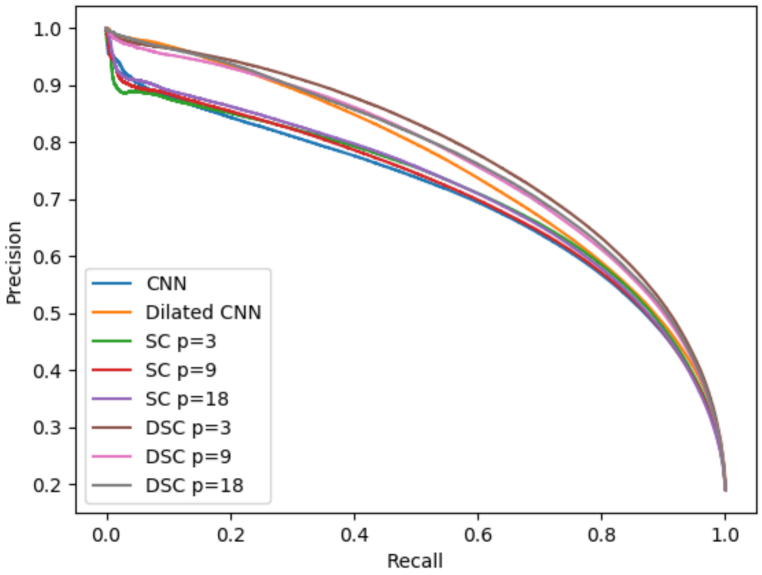


Figure 4.4: Precision-recall curve.

Figure 4.5: Visual comparison. From left to right, the input image, the ground truth, the CNN, the dilated CNN, the SC ($p = 3, 9, 18$), and the DSC ($p = 3, 9, 18$).

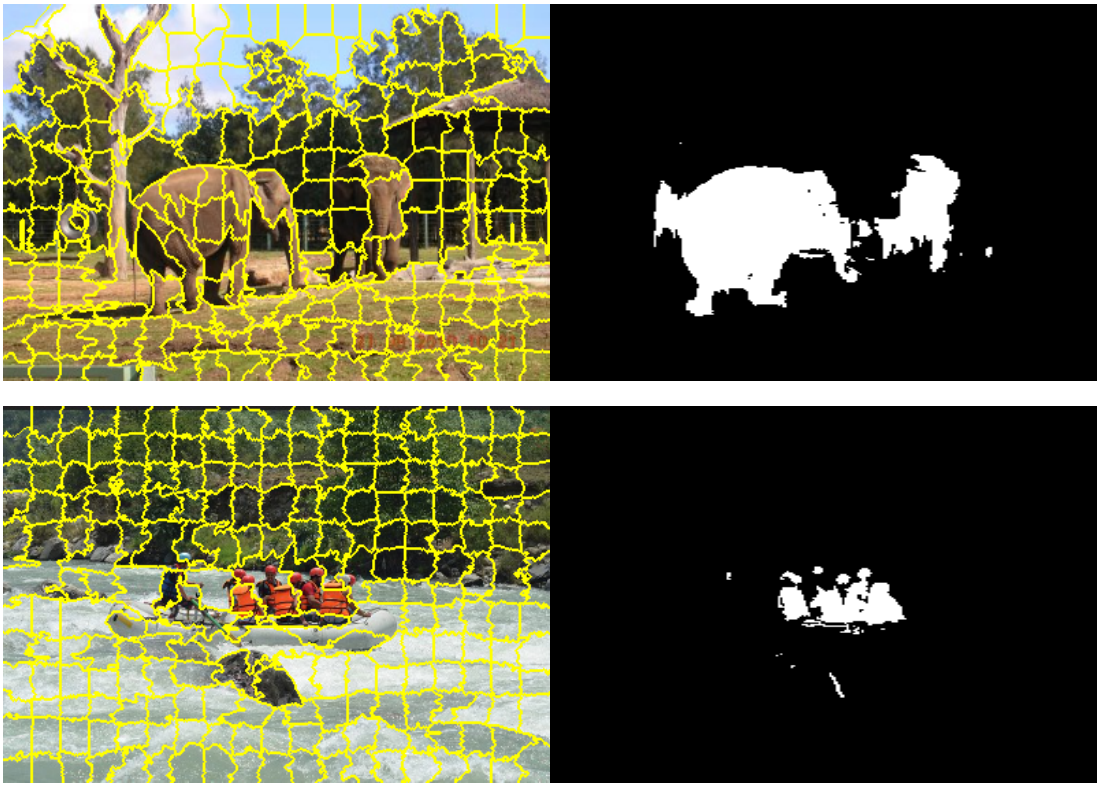


Figure 4.6: Example results of failure cases. SLIC [Achanta et al., 2012] groups the pixels that are similar in color and pixel position. Thus, SLIC cannot separate background and foreground pixels (e.g., the left elephant and the tree, the boat and river).

5

Integrating Superpixel Segmentation into Fully Convolutional Networks

In image segmentation with CNNs, the downsampling layer causes the loss of detailed information, such as high frequency components, object boundaries, and small and/or thin objects. Therefore, the trainable decoder and dilated convolution were proposed to solve it. However, these techniques cause an increase in the trainable parameters and/or the computational costs.

In this chapter, we introduce superpixel segmentation scheme into FCNs [Long et al., 2015] to mitigate the information loss. We show the overview of our proposed method in Figure 5.1. Our approach views the general downsampling as superpixel-based downsampling and groups the pixels by utilizing sampled pixels as superpixel seeds. The coarse prediction map is decoded to a fine resolution based on the generated superpixels.

By integrating our proposed method into downsampling layers, the model hierarchically generates superpixels and predicts the target values for superpixels. The proposed method does not change the feed-forward path of a base architecture because superpixels are not used in the feed-forward path, but they are only used for recovering the lost resolution instead of bilinear interpolation. Be-

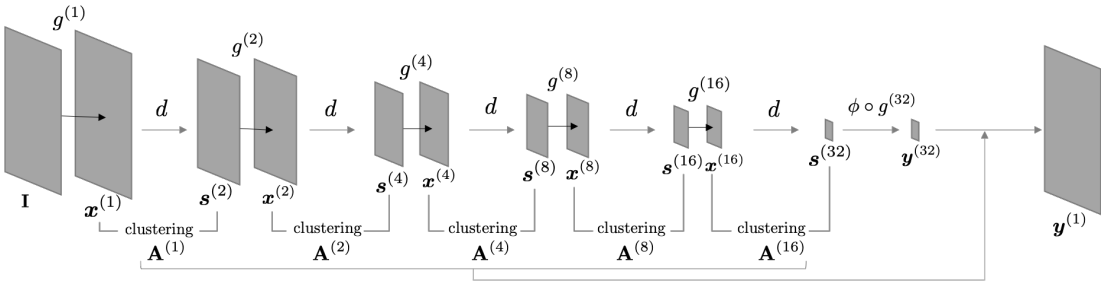


Figure 5.1: Illustration of proposed hierarchical clustering for FCN-32s [Long et al., 2015]. The proposed method groups the pixels in downsampling layers and forms a set of pixels as an assignment matrix. The model predicts the target values for a set of pixels. Unlike existing methods combining superpixel segmentation and neural networks [Kwak et al., 2017, Suzuki et al., 2018, Yang et al., 2020], the proposed method does not use superpixels for downsampling, explicitly. Therefore, the proposed method can be plugged into existing architectures without a change in their feed-forward paths. Superpixels are only used to recover the resolution using eq. (5.3) instead of bilinear interpolation.

cause the superpixels preserve the detailed information, the model mitigates the information loss.

5.1 Preliminaries

Let $\mathbf{I} \in \mathbb{R}^{HW \times 3}$ be an RGB image where H and W denote the image height and width and $\boldsymbol{x}_i^{(s)} \in \mathbb{R}^N$ be a N-dimension feature vector of an i-th pixel in the feature map, where s denotes an output stride; namely, the resolution of $\boldsymbol{x}^{(s)}$ is $(H/s, W/s)$. FCNs consist of the blocks built on convolution layers and ReLU [Krizhevsky et al., 2012] activation and downsampling layers, such as max-pooling and convolution layers with a stride of two or more. Let $g^{(s)}$ and $d^{(s)} : \mathbb{R}^{\frac{HW}{s^2} \times N} \rightarrow \mathbb{R}^{\frac{HW}{s'^2} \times M}$, $s < s'$ be the convolution and ReLU blocks and a downsampling layer that reduces the spatial resolution. For simplicity, we define the downsampling layer as the operation reducing the resolution in half, from $(\frac{H}{s}, \frac{W}{s})$ to $(\frac{H}{2s}, \frac{W}{2s})$, without loss of generality, and we describe the downsampling layer as just $d(\cdot)$ in the rest of this chapter. Note that the feature dimension of the downsampled feature map may change when using the strided convolution as the downsampling. Then, as shown in Figure 5.1, the prediction map generated by FCN-32s [Long et al., 2015] is defined as $\boldsymbol{y}^{(32)} = \phi \circ g^{(32)} \circ d \circ \cdots \circ g^{(2)} \circ d \circ g^{(1)}(\mathbf{I})$, where $\phi(\cdot)$ maps the input feature maps into target values. The proposed method does not change the feed-forward path, namely, the map from $\mathbf{I}$ to $\boldsymbol{y}^{(32)}$.

5.2 Clustering and Upsampling Procedure

Our strategy is to group pixels at downsampling layers, predict the target values for clusters, and share the predicted value with pixels belonging to the corresponding cluster. The j-th cluster at the output stride of s is defined as $c_j^{(s)} = \{i | \forall k, \mathcal{S}(\mathbf{W}^{(s)}\boldsymbol{x}_i^{(s)}, \tilde{\mathbf{W}}^{(2s)}\boldsymbol{s}_j^{(2s)}) \geq \mathcal{S}(\mathbf{W}^{(s)}\boldsymbol{x}_i^{(s)}, \tilde{\mathbf{W}}^{(2s)}\boldsymbol{s}_k^{(2s)})\}$, where $\boldsymbol{s}_j^{(2s)}$ denotes the j-th pixel of the downsampled feature map, $\boldsymbol{s}^{(2s)} = d(\boldsymbol{x}^{(s)})$, and $\mathcal{S}$ denotes a similarity function that is defined as the cosine similarity in our experiments. $\mathbf{W}^{(s)} \in \mathbb{R}^{K \times N}$ and $\tilde{\mathbf{W}}^{(2s)} \in \mathbb{R}^{K \times M}$ are learnable weight matrices for

$\boldsymbol{x}^{(s)} \in \mathbb{R}^{\frac{HW}{s^2} \times N}$ and $\boldsymbol{s}^{(2s)} \in \mathbb{R}^{\frac{HW}{4s^2} \times M}$, which maps the feature vectors into a K-dimension space. K is set to 64 for our experiments. This clustering process is executed in downsampling layers and generates the set of pixels in each resolution, $\{c^{(s)}\}$.

Although the cluster to which an i-th pixel belongs can be obtained by $\arg\max_k \mathcal{S}(\mathbf{W}^{(s)}\boldsymbol{x}_i^{(s)}, \tilde{\mathbf{W}}^{(2s)}\boldsymbol{s}_k^{(2s)})$, it is non-differentiable. Therefore, to train the model in an end-to-end manner, we relax it by using the temperature softmax function. We first formulate hard clustering and next describe its relaxed version.

The hard clustering problem is defined as follows:

$$\mathbf{A}^{(s)*} := \underset{\mathbf{A}^{(s)} \in \{0,1\}^{U \times V}}{\arg\max} \sum_{ij} \mathbf{A}_{ij}^{(s)} \mathbf{S}_{ij}^{(s)}, \ s.t., \ \sum_j \mathbf{A}_{ij}^{(s)} = 1, \tag{5.1}$$

where $\mathbf{A}^{(s)*} \in \{0,1\}^{U \times V}$ and $\mathbf{S}^{(s)} \in \mathbb{R}_+^{U \times V}$ denote an assignment matrix and a non-negative similarity matrix that is defined as $\mathbf{S}_{ij}^{(s)} = \mathcal{S}(\mathbf{W}^{(s)}\boldsymbol{x}_i^{(s)}, \tilde{\mathbf{W}}^{(2s)}\boldsymbol{s}_j^{(2s)})$. U and V denote the number of pixels in $\boldsymbol{x}^{(s)}$, i.e., $U = \frac{HW}{s^2}$ and $V = \frac{HW}{4s^2}$, and $\boldsymbol{s}^{(2s)}$, respectively. We illustrate the maximization problem in Fig. 5.2 and obtained superpixels in Fig. 5.3.

Then, we can decode the downsampled feature map to the fine resolution feature map based on the clusters as follows:

$$\tilde{\boldsymbol{x}}^{(s)} := \mathbf{A}^{(s)*}\boldsymbol{x}^{(2s)}. \tag{5.2}$$

The model predicts target values from the coarsest feature map, $\boldsymbol{y}^{(32)} = \phi(\boldsymbol{x}^{(32)})$, and then the coarsest prediction can be decoded to the original resolution by recursively decoding $\boldsymbol{y}^{(32)}$ as follows:

$$\boldsymbol{y}^{(1)} = \prod_{s'=\{16,8,4,2,1\}} \mathbf{A}^{(s')*}\boldsymbol{y}^{(32)}. \tag{5.3}$$

Note that we plug the clustering modules into a part of all the downsampling layers in our experiments. Therefore, we decode the prediction with the bilinear upsampling to the original resolution after decoding it to the plausible fine resolution using eq. (5.3).

Unfortunately, the clustering procedure is non-differentiable because of $\arg\max$ in eq. (5.1). Thus, we relax the assignment matrix $\mathbf{A}^{(s)*}$ to a soft

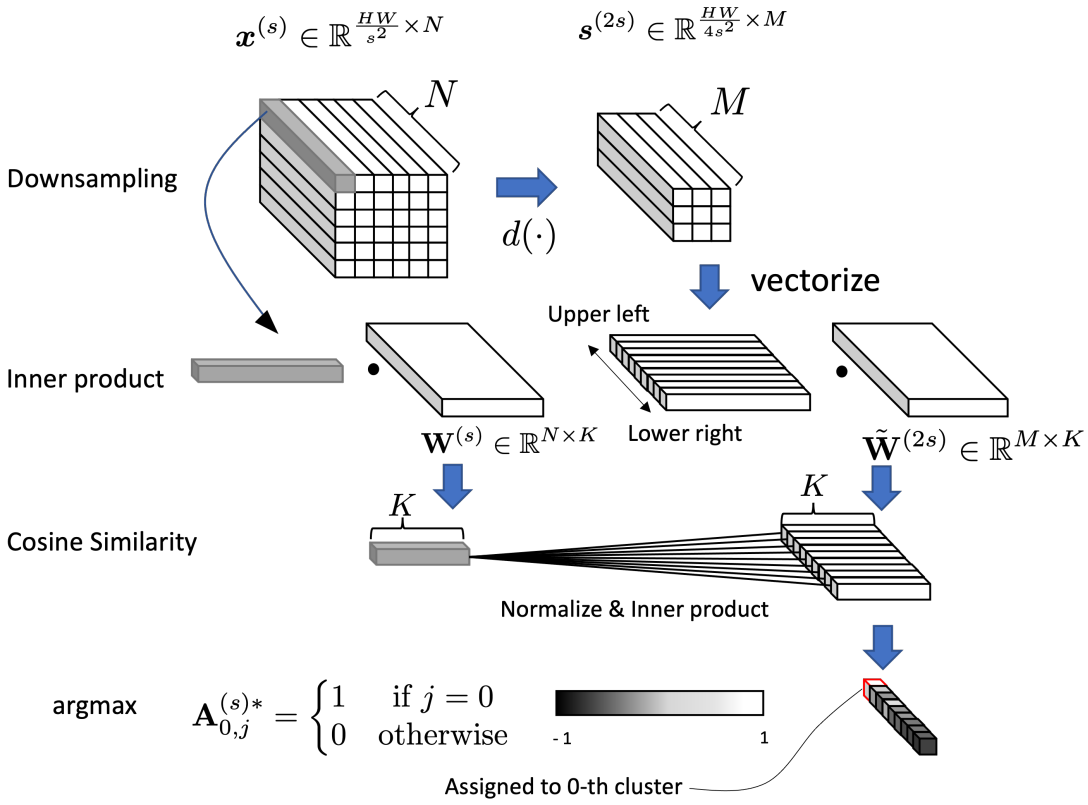


Figure 5.2: Illustration of the clustering procedure represented in eq. (5.1). Given the feature maps before and after downsampling, we first compute the inner product between each pixel in the feature maps and the trainable weight matrix. Then, we compute the similarity matrix with the cosine similarity between the pixels in the feature maps before and after downsampling. Finally, we compute the assignment matrix $\mathbf{A}^{(s)*}$, where $\mathbf{A}^{(s)*}_{ij} = 1$ if the j-th pixel in the downsampled feature map has maximum similarity for the i-th pixel in the feature map before downsampling, and 0 otherwise.

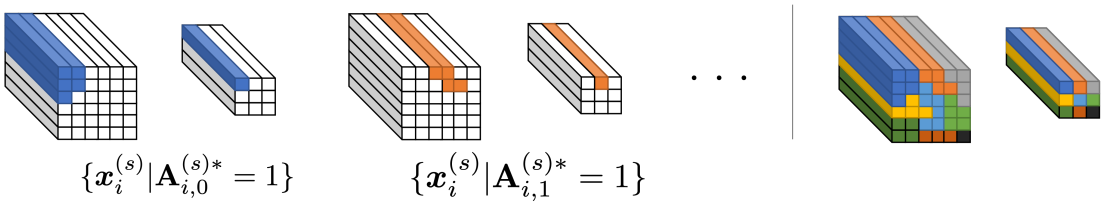


Figure 5.3: Illustration of the computed cluster, i.e., superpixels. We visualize a set of pixels and their corresponding cluster seeds in the same color.

assignment matrix $\tilde{\mathbf{A}}^{(s)} \in (0,1)^{U \times V}$. We define the soft assignment from $\boldsymbol{x}_i^{(s)}$ to the sampled cluster seed $\boldsymbol{s}_j^{(2s)}$ as follows:

$$\tilde{\mathbf{A}}_{ij}^{(s)} := \frac{\exp\left(\mathcal{S}\left(\mathbf{W}^{(s)}\boldsymbol{x}_i^{(s)}, \tilde{\mathbf{W}}^{(2s)}\boldsymbol{s}_j^{(2s)}\right)/\tau\right)}{\sum_k \exp\left(\mathcal{S}\left(\mathbf{W}^{(s)}\boldsymbol{x}_i^{(s)}, \tilde{\mathbf{W}}^{(2s)}\boldsymbol{s}_k^{(2s)}\right)/\tau\right)}, \tag{5.4}$$

where τ is a temperature parameter. If $\tau \to 0$, $\tilde{\mathbf{A}}^{(s)}$is equal to $\mathbf{A}^{(s)*}$. We set τ to 0.07 for our experiments. The dense prediction map is generated using eq. (5.3) and $\tilde{\mathbf{A}}^{(s)}$ instead of $\mathbf{A}^{(s)*}$. When calculating the loss between the ground truth labels and the prediction map generated using eq. (5.3) with $\tilde{\mathbf{A}}^{(s)}$, the loss is fully backpropable, and the model can be trained in an end-to-end manner.

We visualize hierarchical clustering in Figure 5.4. The clusters are generated by the FCN-32s with our proposed method trained on Cityscapes [Cordts et al., 2016]. The pixels are hierarchically grouped, and the model forms $\frac{HW}{32^2}$ clusters in the end. The clusters preserve the object boundaries and the small and/or thin objects, such as signs and poles.

5.3 Practical Issues and Their Solutions

The soft assignment matrix is not computationally applicable; for example, when the resolution of an input image is 512×512 and downsampled to 256×256, the soft assignment matrix requires 64GByte in the single-precision floating-point number. Therefore, we restrict the number of candidate clusters to only nine surrounding clusters, as shown in Figure 5.5, and bring

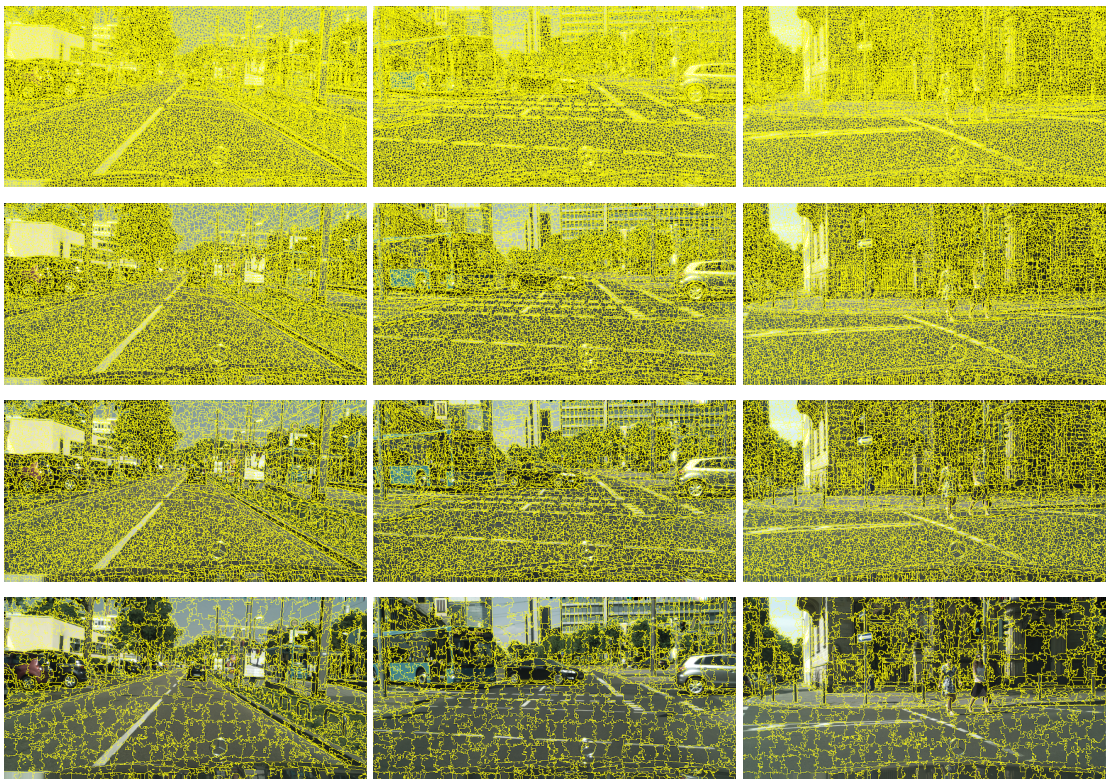


Figure 5.4: An example result of hierarchical clustering. From top to bottom, $c^{(2)}$, $c^{(4)}$, $c^{(8)}$, $c^{(16)}$. We visualize cluster boundaries as yellow lines. These clusters are generated by FCN-32s with the ResNet-101 backbone combined with our proposed method. Note that we visualize the cluster boundaries; hence, the pixels may be assigned in the same cluster even if they are separated by the yellow line.

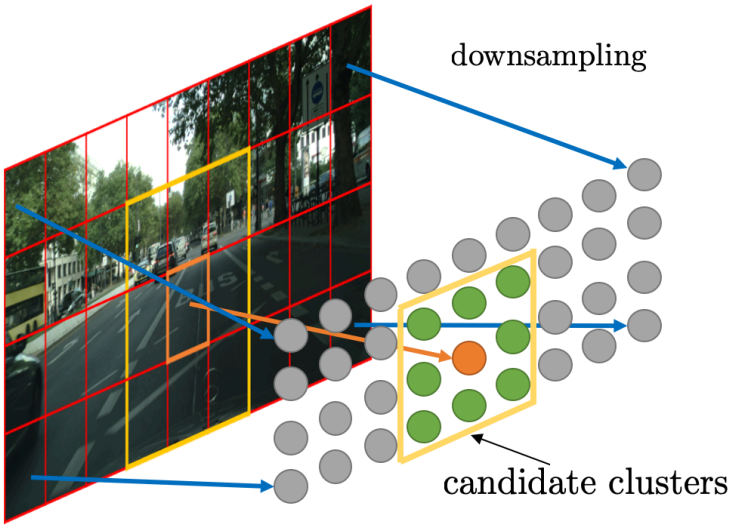


Figure 5.5: Illustration of the candidate clusters. Grid is defined as a stride, meaning that if a stride is two, then each cell has 2×2 pixels. Circles indicate the cluster seeds sampled from corresponding regions. The candidate clusters to which pixels in the orange cell belong are defined as the orange circle and eight neighbors.

down the size of $\tilde{\mathbf{A}}^{(s)}$ from $\frac{HW}{s^2} \times \frac{HW}{4s^2}$ to $\frac{HW}{s^2} \times 9$ elements. The similar or same technique is also used in existing superpixel segmentation methods [Achanta et al., 2012, Jampani et al., 2018, Yang et al., 2020].

The flexibility of the downsampling operation is important for the clustering since the downsampled pixels are used as the cluster seeds. However, static downsampling operations, such as max-pooling and strided convolution, may not sample effective seeds for clustering because they simply sample pixels from a fixed region. Therefore, we use the modulated deformable convolution (DCNv2) [Zhu et al., 2019] with a stride of two as the downsampling operation, which can adaptively change kernel shapes and weights by the learnable offsets and modulation parameter generator. We verify the effectiveness of DCNv2 for our proposed method in the next section.

5.4 Evaluation

We first evaluate the hierarchical clustering module proposed in Chapter 5 on the dense prediction tasks, such as semantic segmentation, superpixel segmentation, and monocular depth estimation.

For semantic segmentation, we first evaluate the proposed method with a Cityscapes [Cordts et al., 2016] validation set with various settings as an ablation study and then integrate the proposed method into modern architectures and evaluate them in terms of semantic segmentation, superpixel segmentation, and depth estimation tasks. For a fair comparison, we reproduced all the baseline models in PyTorch [Paszke et al., 2019]. We use "HC" as a prefix to indicate the model using the proposed clustering module. Moreover, we use OS to denote the output stride, meaning that the ratio of the spatial resolution of the input image to the resolution of the prediction map.

5.4.1 Ablation study

In this ablation study, we validate the efficacy and efficiency of our proposed method. We use FCN-32s [Long et al., 2015] with the ResNet backbone [He et al., 2016a] as a base architecture. We train the models with fine training data in Cityscapes to minimize the cross-entropy loss. We set the initial learning rate to 0.01 and decay it with a "poly" learning rate policy where the initial learning rate is multiplied by $(1 - \frac{iter}{max\_iter})^{0.9}$. We employ momentum SGD as an optimizer and set the momentum to 0.9. We train the models for 50K iterations with a batch size of 16. We use random crop with a crop size of 1024×1024, random resize between 0.5 and 2.0, and random horizontal flip as the data augmentation. Moreover, we use the same auxiliary loss as PSPNet [Zhao et al., 2017].

We first evaluate the soft clustering module with various sampling branches. Our module utilizes downsampled feature maps, but the downsampling operations in ResNet are embedded into the building blocks, and we have some choices for obtaining the downsampled feature maps. The building blocks, including downsampling, are defined as $B(\boldsymbol{x}) = I(\boldsymbol{x}) + R(\boldsymbol{x})$, where $I(\cdot)$ denotes a linear

projection called identity mapping and $R(\cdot)$ denotes residual mapping. We compare $I(\boldsymbol{x})$, $R(\boldsymbol{x})$, and $B(\boldsymbol{x})$ as downsampling branches. We show the results in Table 5.1. Note that the soft clustering modules are plugged into conv4_x and conv5_x in ResNet [He et al., 2016a]. The feature map obtained from identity mapping is slightly worse than the others. The residual mapping and the block are superior or inferior depending on the architecture, but the difference between them is less than when comparing them to the identity mapping. We use the block as the sampling branch in the remainder of the experiment.

We next evaluate the mIoU for the proposed method with various hierarchical levels. We increase the number of the proposed modules from conv5_x to conv2_x in the backbone ResNet of the FCN-32s and report their mIoU. We show the results in Figure 5.6. AtrousFCN is a model replacing the striding of conv4_x and conv5_x in the backbone of FCN-32s with atrous convolution. Similar to ours, AtrousFCN preserves the detailed information without a change in the feed-forward path. Our proposed method with one or two clustering modules outperforms AtrousFCN and further improves the mIoU by increasing the hierarchical level. Moreover, the proposed method with ResNet-18 shows higher mIoU than AtrousFCN with ResNet-50 when the hierarchical level is two or more.

We then show frames per second (fps) in Figure 5.7. Our proposed method is two times or more faster than AtrousFCN. Moreover, the proposed method with the hierarchical level of two demonstrates higher mIoU than AtrousFCN, sacrificing only 10% fps compared with FCN-32s. The proposed method with ResNet-18 significantly decreases fps when the hierarchical level is four because the model wastes the inference time for the clustering in the fine resolution feature map.

In our experiments, we use DCNv2 [Zhu et al., 2019] as a downsampling operation because we consider that the flexibility of the downsampling operation is important to sample the effective cluster centers, as described in Chapter 5. To verify it, we finally compare the strided convolution originally used in ResNet with DCNv2 with a stride of two. For FCN-32s, we replace the strided convolutions except for `conv1` in ResNet with DCNv2 with a stride of two.

The results is shown in Figure 5.8. For FCN-32s with a ResNet-18 backbone, DCNv2 significantly improves the mIoU compared with the strided convolution,

Table 5.1: Results on various downsampling branches.

sampling branch	backbone	mIoU
identity	ResNet-18	71.21
	ResNet-50	72.26
residual	ResNet-18	72.43
	ResNet-50	**73.20**
block	ResNet-18	**72.70**
	ResNet-50	72.93

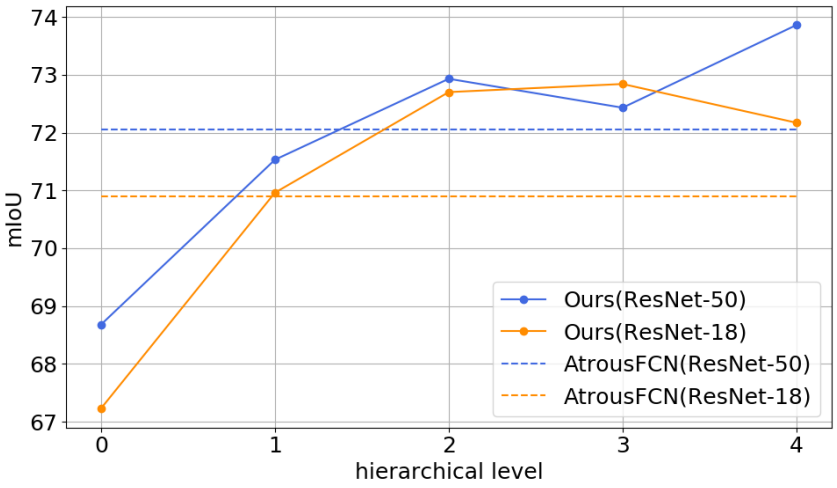


Figure 5.6: The mIoU for various hierarchical levels for ResNet-18 and ResNet-50 as the backbone. The model corresponds to the FCN-32 [Long et al., 2015] when the level is zero. Our proposed method with a level of two outperforms AtrousFCN for both backbones.

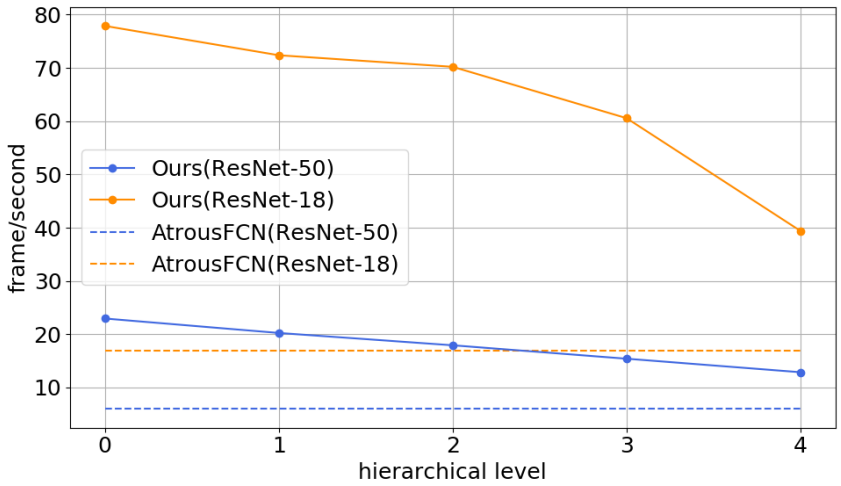


Figure 5.7: Inference speed for various hierarchical levels for ResNet-18 and ResNet-50 as the backbone. We report average fps over 100 trials for a 1024×2048 input. Our proposed method is two times or more faster than AtrousFCN.

but for ResNet-50, DCNv2 slightly decreases the mIoU compared to the strided convolution. For the proposed method, DCNv2 stably improves the mIoU. Although DCNv2 is better than the strided convolution, the proposed method with the strided convolution outperforms AtrousFCN when the hierarchical level is two or more. AtrousFCN shows 70.90 and 72.06 mIoU on ResNet-18 and ResNet-50, respectively, and the proposed method using the strided convolution shows 71.17 and 72.37 mIoU on ResNet-18 and ResNet-50, respectively, when the level is two. Thus, DCNv2 is a better choice for the proposed method, but the strided convolution also works effectively.

5.4.2 Semantic Segmentation

We compare modern architectures, FCN-32 [Long et al., 2015], FCN with atrous convolution (AtrousFCN) [Chen et al., 2014, Yu and Koltun, 2015], FPN [Lin et al., 2017], PSPNet [Zhao et al., 2017], and DeepLabv3 [Chen et al., 2017b], to the models adopting soft clustering

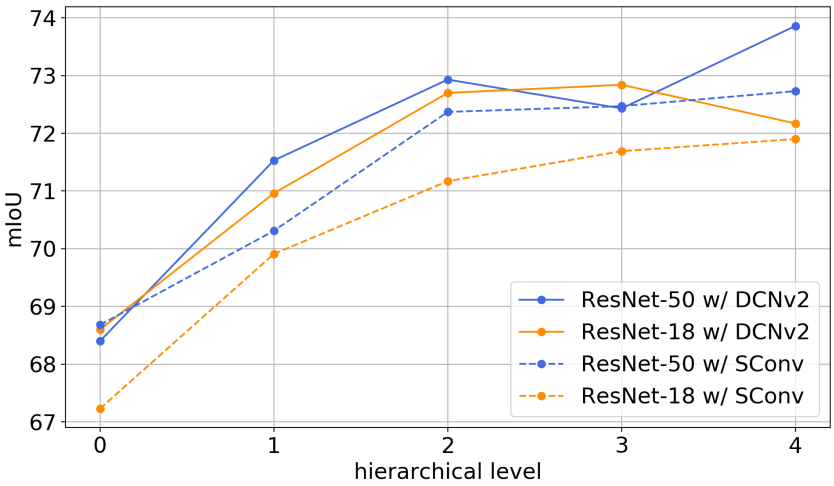


Figure 5.8: Comparison of the strided convolution (SConv) and DCNv2 as the downsampling layer. The hierarchical level of zero corresponds to the FCN-32s [Long et al., 2015].

on the Cityscapes [Cordts et al., 2016] test set. The FCN-32s is the simplest model for dense prediction, and AtrousFCN is a detail-preserving model. FPN is an asymmetric encoder-decoder model proposed for object detection, but it has also been used for a panoptic segmentation task [Kirillov et al., 2019a]. PSPNet and DeepLabv3 are AtrousFCN combined with the spatial pyramid pooling. The training protocol is the same as the ablation study. We use ResNet-101 [He et al., 2016a] for the backbone architecture. The "HC" models use the clustering modules for conv4_x and conv5_x in ResNet-101.

We show the mIoU in Table 5.2 and the example results in Figure 5.9. Note that we evaluate the models with a multi-scale input and use the average results following [Zhao et al., 2017]. HCFCN-32s shows better mIoU than the FCN-32. Particularly, HCFCN-32s improves mIoU for thin and small objects (e.g., poles, traffic lights, and signs) that are often missed in the models that generate low-resolution maps. In fact, HCFCN-32s predicts such objects better than the FCN-32, as shown in Figure 5.9. Our proposed method also improves the mIoU for FPN, namely, the encoder-decoder model. The HCFPN also enhances 1%-2% of the mIoU for small objects from FPN.

Table 5.2: Per-class results on the Cityscapes test set. We use ResNet-101 as the backbone and evaluate the models with a multi-scale input following [Zhao et al., 2017].

Model	OS	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
FCN-32s	32	98.4	84.6	91.4	44.6	55.0	49.6	65.3	70.9	91.8	70.7	94.2	81.0	64.2	94.7	61.6	72.4	57.4	65.9	71.5	72.9
AtrousFCN	8	98.5	85.6	92.6	47.1	54.9	66.9	74.1	77.8	93.2	72.0	95.5	85.7	70.5	95.4	58.7	72.0	61.4	69.3	75.4	76.1
FPN	8	98.5	85.5	92.4	45.5	55.6	63.8	71.4	76.1	93.0	71.4	95.4	84.6	67.5	95.4	63.9	72.2	57.7	67.4	74.5	75.4
PSPNet	8	98.6	86.0	92.9	50.2	56.7	66.9	74.0	78.0	93.3	72.0	95.7	85.7	70.3	95.7	68.2	82.2	74.2	68.8	75.7	78.2
DeepLabv3	8	98.6	86.5	93.0	48.5	57.1	67.5	74.4	78.4	93.3	72.5	95.6	86.0	72.0	95.7	71.4	82.4	74.3	69.7	75.8	78.6
HCFCN-32s	32	98.5	85.4	92.5	47.1	55.4	63.5	73.3	77.2	93.2	71.2	95.3	85.1	68.9	95.5	61.9	72.1	58.4	68.1	75.1	75.7
HCFPN	8	98.5	85.2	92.5	47.4	55.4	64.7	73.4	77.1	93.2	71.2	95. 1	85.2	69.6	95.6	63.3	72.3	58.6	69.3	75.3	76.0
HCPSPNet	32	98.4	84.8	92.7	52.1	56.8	62.9	72.9	76.9	93.2	71.4	95.3	85.1	70.1	95.7	70.7	80.3	70.1	69.3	75.2	77.6
HCDeepLabv3	32	98.6	85.7	92.7	52.2	55.6	62.8	73.0	76.9	93.2	71.7	95.4	85.0	69.8	95.6	69.2	81.3	72.8	68.2	74.7	77.6

Figure 5.9: Example results on the Cityscapes validation set. From left to right, (a) the input, (b) the ground truth, (c) FCN-32, (d) HCFCN-32, (e) PSPNet, and (f) HCPSPNet. Each model uses ResNet-101 as the backbone. HCFCN-32s and HCPSPNet integrate the clustering modules into conv4_x and conv5_x in ResNet.

However, HCPSPNet and HCDeepLabv3 show slightly lower mIoU than their original models. HCPSPNet and HCDeepLabv3 result in a significant drop in the mIoU for small and/or thin objects such as poles and signs. We consider that the pyramid pooling module in PSPNet and DeepLabv3 impairs the detailed information for HCPSPNet and HCDeepLabv3 because their mIoU value for small objects is lower than the HCFCN-32. The pyramid pooling modules are useful in capturing global information and enhancing mIoU for large objects. In fact, PSPNet and DeepLabv3 significantly improve mIoU for large objects compared with AtrousFCN. HCPSPNet and HCDeepLabv3 also significantly improve mIoU for the large objects but degrade about 1% of the mIoU for small objects compared with the HCFCN-32. Thus, the global context aggregation for low-resolution feature maps may impair the local context.

AtrousFCN shows slightly higher the mIoU than the HCFCN-32, although HCFCN-32s is better when the backbone is ResNet-18 and ResNet-50. This fact may imply that our proposed method stably works, regardless of model size, but AtrousFCN with a shallow backbone is trapped in worse local minima.

We show the accuracy and inference time in Table 5.3. We report the average time over 100 trials on the NVIDIA Quadro RTX8000 GPU with a 1024×2048 input and mIoU on the validation set. Our proposed method significantly reduces latency for PSPNet and DeepLabv3 and improves the mIoU for the FPN and FCN-32s with a small increase in the inference time.

We also evaluate our proposed method using ADE20K [Zhou et al., 2016]. ADE20K contains a broader range of scene and object categories than Cityscapes. We train the models for 125k iterations with the same protocol as the ablation study except that we set the initial learning rate to 0.02. As shown in Table 5.4, HCPSPNet and HCDeepLabv3 show comparable results to PSPNet and DeepLabv3, but the proposed methods are significantly faster.

5.4.3 Superpixel Segmentation

We evaluate soft clustering for superpixel segmentation on the BSDS500 [Arbelaez et al., 2010] test set. The BSDS500 contains 200 training images, 100 validation images, and 200 test images, and we use the training set

Table 5.3: Accuracy and inference times of various architectures for a 1024×2048 input on the Cityscapes validation set. The inference time is the average time over 100 trials. We evaluate the models with a single-scale input.

Method	mIoU	Pixel acc.	msec/image
FCN-32s	71.7	94.9	71.7
AtrousFCN	76.3	96.0	296.7
HCFCN-32s	76.0	96.1	85.9
FPN	75.1	95.8	82.0
HCFPN	77.2	96.2	92.9
PSPNet	77.7	96.2	298.7
HCPSPNet	77.6	96.2	87.0
DeepLabv3	78.4	96.3	380.8
HCDeepLabv3	77.4	96.2	93.2

for training. We employ the SSN [Jampani et al., 2018] as a baseline method that is a supervised superpixel segmentation method that consists of an original shallow encoder-decoder model. We introduce soft clustering into the SSN and replace bilinear upsampling operations with Eq. (5.3).

We evaluate these models using ASA and BR. ASA quantifies the achievable accuracy for given segmentation labels using superpixels as a preprocessing step and BR assesses boundary adherence given ground truth labels [Stutz et al., 2018].

We set the initial learning rate to 5e-5 and decay it with a cosine learning rate policy to 5e-7. We employ the Adam optimizer [Kingma and Ba, 2014] as an optimizer and train models for 300K iterations with a batch size of 8. We use random crop with a crop size of 208×208 and random horizontal flip as the data augmentation. The pixel feature dimension is set to 20. The number of superpixels and differentiable SLIC iterations are set to 169 and 5 during training, respectively. For testing, we set the iterations to 10.

Table 5.4: Results on the ADE20K validation set. The inference time is the average time over 100 trials on the NVIDIA Quadro RTX8000 with a 512×512 input. We use ResNet-101 as the backbone architecture.

Models	mIoU	Pixel Acc.	msec/image
PSPNet	42.53	80.91	48.74
HCPSPNet	42.56	80.85	18.83
DeepLabv3	42.70	81.04	62.51
HCDeepLabv3	42.71	81.07	20.49

We show ASA and BR in Figures 5.10 and 5.11 and the example results in Figure 5.12. The hierarchical soft clustering improves both ASA and BR, and as shown in Figure 5.12, we can observe that the HCSSN reduces the undersegmentation error, which is a metric resembling $(1 - \text{ASA})$ [Stutz et al., 2018].

5.4.4 Monocular Depth Estimation

Our proposed method imposes local smoothness because of the clustering-based upsampling procedure, which may hurt the accuracy of the regression tasks. Therefore, we also evaluate the proposed method for monocular depth estimation using NYU Depth v2 [Silberman et al., 2012].

As a baseline method, we use the deep ordinal regression network (DORN) [Fu et al., 2018], which consists of a dense feature extractor and a scene understanding module. We compare DORN to a model that replaces the atrous convolution in the dense feature extractor with striding and that uses our soft clustering. Note that our model decodes a feature map generated by the scene understanding module with Eq. (5.3), although the segmentation models decode the prediction map.

We train and evaluate the models with the same protocol as [Fu et al., 2018]. We report the average inference time over 100 trials on the NVIDIA Quadro RTX8000 with a 257×353 input.

We show the comparison results in Table 5.5. Our soft clustering also produces

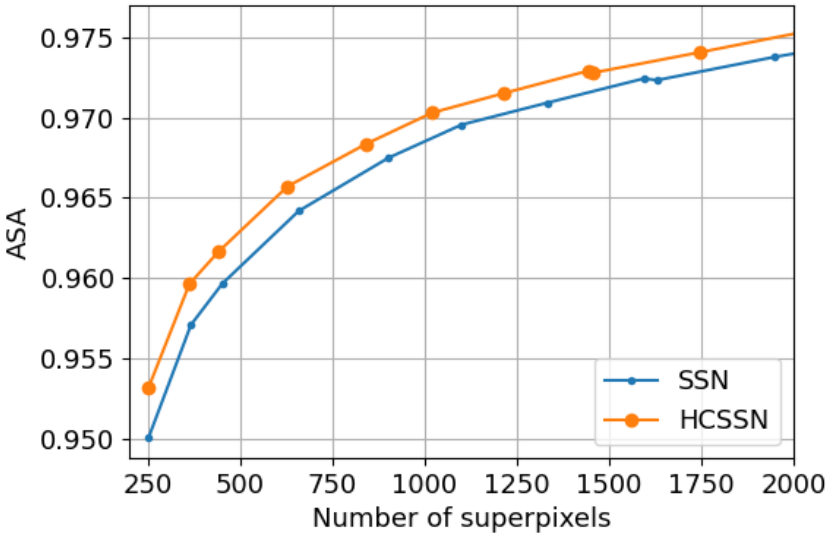


Figure 5.10: Achievable segmentation accuracy on the BSDS500 [Arbelaez et al., 2010].

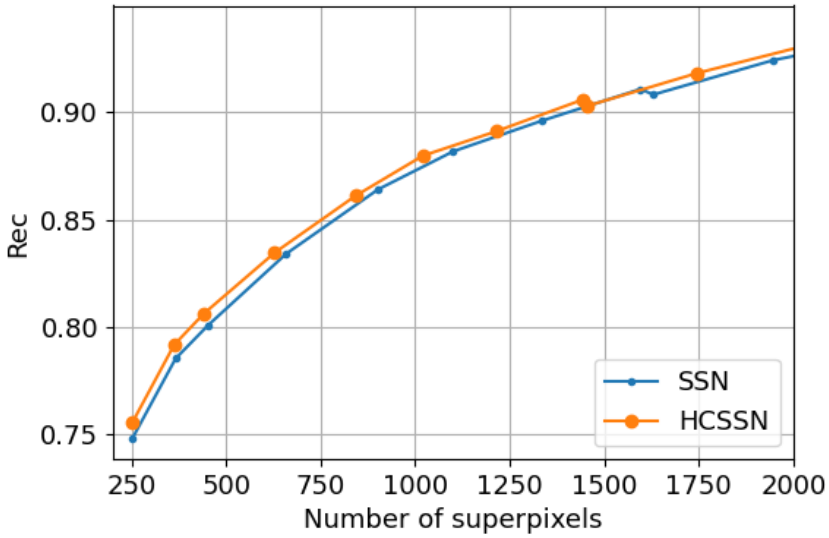


Figure 5.11: Boundary recall on the BSDS500 [Arbelaez et al., 2010].

Figure 5.12: Example results of the HCSSN (left) and the SSN (right). Red regions denote the undersegmentation error [Stutz et al., 2018] that measures the "leakage" of superpixels with respect to ground truth labels. Thus, the images with fewer red areas are better.

Model	δ_1	δ_2	δ_3	rel	$\log_{10}$	rms	msec/image
DORN [Fu et al., 2018]	0.83	0.97	0.99	0.12	0.05	0.51	39.1
HCDORN	0.84	0.97	0.99	0.11	0.05	0.50	20.3

Table 5.5: Results on NYU Depth v2 [Silberman et al., 2012]. HCDORN shows the same accuracies with fewer computational time than DORN.

a speedup without impairing the accuracy of the monocular depth estimation.

5.5 Conclusion

We proposed a superpixel segmentation scheme integrated into FCNs. The proposed method mitigates information loss due to downsampling layers by preserving detailed information in the form of superpixels and using them to upsampling images. As a result, we improve the segmentation accuracy of the FCN-32 architectures and the inference speed of the AtrousFCN architectures without accuracy degradation. Moreover, we verify that the proposed method also works well on depth estimations, namely, regression tasks.

6

Conclusion

6.1 Thesis Summary

Image segmentation forms the foundation for image-based systems, such as autonomous driving, advanced driver-assistance, and surveillance systems. CNNs have brought great progress in the image processing and computer vision fields. Then, many CNN-based segmentation methods have been proposed and have demonstrated attractive performances. However, CNNs often miss detailed information, such as high-frequency components and object boundaries, due to the downsampling operations. Therefore, some techniques have been introduced to CNNs: the trainable decoder is used to recover the lost spatial resolution, and the atrous convolution is used instead of the stride of downsampling. These techniques are useful, but they do not directly tackle the problem of downsampling operations.

In this thesis, we first proposed a CNN-based superpixel segmentation method. The method generates superpixels from a single unlabeled image via a CNN by maximizing mutual information. Through the method, We investigated the CNNs' prior for superpixels. In practice, our proposed method shows better ASA and BR than baseline methods, when the number of superpixels is small. From the comparison results, we determine that the structure of CNNs has better prior to segment pixels than the hand-crafted color- and position-based objective.

We next proposed the superpixel convolution and superpixel dilated convolution. These convolution operations are designed for superpixel images and the generalization of convolution operations on image domains. We built a superpixel-based segmentation architecture through superpixel pooling and our proposed convolution operations and compared it with general CNNs. In the foreground segmentation task, our superpixel-based segmentation method demonstrated effectiveness and outperformed the general CNNs. The results indicate that the detailed information preservation by superpixels improved the segmentation accuracy for the CNNs.

Both the proposed superpixel segmentation and the superpixel convolution provided good insight suggesting that the superpixels could potentially improve the CNNs' performance. However, they were used independently; in other words, the superpixel segmentation and image segmentation schemes were separated. Thus, we finally proposed the end-to-end framework, which integrates superpixel segmentation into general CNNs. Our proposed method views the general downsampling operation as a centroid sampling operation and implicitly generates superpixels at downsampling layers. The method demonstrated its efficacy and efficiency compared to the trainable decoder and the atrous convolutions on several tasks. Our proposed method can introduce existing CNN architectures without a change in their feed-forward path. Therefore, we expect to apply it to a wide range of applications.

6.2 Future work

Generated superpixels by our proposed superpixel segmentation method depends on the initialization of CNN parameters. Thus, the generated superpixels will be different even if the same input and the same CNN architecture are used. Depending on the application, this property may be a drawback. However, it also leads to an extension to Bayesian inference and an understanding of the image. For example, one can obtain several results from different initial parameters and then computes the variance. One can use the variance for segmentation to refine the results. Moreover, the regions having large variance may denote the

essential difficulty of segmentation. Therefore, we consider the extension to be an important direction.

Our proposed end-to-end framework generates clusters in the back end. This scheme may be able to simplify instance and panoptic segmentation frameworks. The recently developed instance and panoptic segmentation frameworks are slightly complex. For example, Mask R-CNN [He et al., 2017] first generates object candidates and then predicts bounding boxes, class labels, and instance masks. The framework is generally called a two-stage detector and is known to be complex to implement. PanopticFPN [Kirillov et al., 2019a], which is a panoptic segmentation method, is also based on a two-stage framework. If our proposed method can cluster the pixels for each instance, the instance/panoptic segmentation framework may be realized with the same framework as semantic segmentaion. As shown in Figure 5.4, the generated cluster fits the instances, although oversegmented. If we can generate super nodes that represent the instance and cluster the segments, the instance/panoptic segmentation can be implemented as a node-wise classification model similar to the CNN-based semantic segmentation scheme.

References

[Achanta et al., 2012] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

[Arbelaez et al., 2010] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[Badrinarayanan et al., 2017] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495.

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

[Boykov and Jolly, 2001] Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *International Conference on Computer Vision*, volume 1, pages 105–112. IEEE.

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

REFERENCES

[Bridle et al., 1992] Bridle, J. S., Heading, A. J., and MacKay, D. J. (1992). Unsupervised classifiers, mutual information and'phantom targets. In *Advances in Neural Information Processing Systems*, pages 1096–1101.

[Bruna et al., 2013] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. Technical report.

[Chen et al., 2014] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. Technical report.

[Chen et al., 2017a] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848.

[Chen et al., 2017b] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017b). Rethinking atrous convolution for semantic image segmentation. Technical report.

[Chen et al., 2018] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision*, pages 801–818.

[Ciresan et al., 2012] Ciresan, D., Giusti, A., Gambardella, L., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in Neural Information Processing Systems*, 25:2843–2851.

[Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223.

[Defferrard et al., 2016] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

[Dosovitskiy et al., 2015] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2758–2766.

[Farabet et al., 2012] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2012). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.

[Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

[Fu et al., 2018] Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011.

[Gadde et al., 2016] Gadde, R., Jampani, V., Kiefel, M., Kappler, D., and Gehler, P. V. (2016). Superpixel convolutional networks using bilateral inceptions. In *European Conference on Computer Vision*, pages 597–613. Springer.

[Godard et al., 2017] Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279.

[Godard et al., 2019] Godard, C., Mac Aodha, O., Firman, M., and Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3838.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27:2672–2680.

[Gould et al., 2009] Gould, S., Fulton, R., and Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision*, pages 1–8. IEEE.

REFERENCES

[Grady, 2006] Grady, L. (2006). Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783.

[Gupta et al., 2013] Gupta, S., Arbelaez, P., and Malik, J. (2013). Perceptual organization and recognition of indoor scenes from RGB-D images. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571.

[Hammond et al., 2011] Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *International Conference on Computer Vision*, pages 2961–2969.

[He et al., 2012] He, K., Sun, J., and Tang, X. (2012). Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409.

[He et al., 2016a] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer.

[He et al., 2004] He, X., Zemel, R. S., and Carreira-Perpiñán, M. Á. (2004). Multiscale conditional random fields for image labeling. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–II. IEEE.

[He et al., 2006] He, X., Zemel, R. S., and Ray, D. (2006). Learning and incorporating top-down cues in image segmentation. In *European Conference on Computer Vision*, pages 338–351. Springer.

[Hoogeboom et al., 2019] Hoogeboom, E., Berg, R. v. d., and Welling, M. (2019). Emerging convolutions for generative normalizing flows. *arXiv preprint arXiv:1901.11137*.

[Iizuka et al., 2017] Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):1–14.

[Ilg et al., 2017] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Technical report.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134.

[Jampani et al., 2018] Jampani, V., Sun, D., Liu, M.-Y., Yang, M.-H., and Kautz, J. (2018). Superpixel sampling networks. In *European Conference on Computer Vision*, pages 352–368. Springer.

[Johnson et al., 2018] Johnson, J., Gupta, A., and Fei-Fei, L. (2018). Image generation from scene graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1219–1228.

[Kanezaki, 2018] Kanezaki, A. (2018). Unsupervised image segmentation by backpropagation. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1543–1547. IEEE.

[Kendall et al., 2015] Kendall, A., Badrinarayanan, V., and Cipolla, R. (2015). Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. Technical report.

[Kirillov et al., 2019a] Kirillov, A., Girshick, R., He, K., and Dollár, P. (2019a). Panoptic feature pyramid networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408.

[Kirillov et al., 2019b] Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P. (2019b). Panoptic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 9404–9413.

[Knyazev et al., 2019] Knyazev, B., Lin, X., Amer, M. R., and Taylor, G. W. (2019). Image classification with hierarchical multigraph networks. Technical report.

[Kohli et al., 2009] Kohli, P., Torr, P. H., et al. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324.

[Krähenbühl and Koltun, 2011] Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in Neural Information Processing Systems*, 24:109–117.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[Kwak et al., 2017] Kwak, S., Hong, S., and Han, B. (2017). Weakly supervised semantic segmentation using superpixel pooling network. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Leung and Malik, 2001] Leung, T. and Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44.

[Levin et al., 2007] Levin, A., Lischinski, D., and Weiss, Y. (2007). A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242.

[Li and Yu, 2015] Li, G. and Yu, Y. (2015). Visual saliency based on multiscale deep features. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 5455–5463.

[Li et al., 2020] Li, X., You, A., Zhu, Z., Zhao, H., Yang, M., Yang, K., and Tong, Y. (2020). Semantic flow for fast and accurate scene parsing. Technical report.

[Lin et al., 2017] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

[Liu et al., 2011] Liu, M.-Y., Tuzel, O., Ramalingam, S., and Chellappa, R. (2011). Entropy rate superpixel segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2104. IEEE.

[Liu et al., 2016] Liu, Y.-J., Yu, C.-C., Yu, M.-J., and He, Y. (2016). Manifold slic: A fast method to compute content-sensitive superpixels. In *Tthe IEEE Conference on Computer Vision and Pattern Recognition*, pages 651–659.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[Matsuo and Aoki, 2015] Matsuo, K. and Aoki, Y. (2015). Depth image enhancement using local tangent plane approximations. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3574–3583.

[McCallum and Sutton, 2005] McCallum, A. and Sutton, C. (2005). Piecewise training of undirected models. In *Conference on Uncertainty in Artificial Intelligence*.

[Mester et al., 2011] Mester, R., Conrad, C., and Guevara, A. (2011). Multichannel segmentation using contour relaxation: fast super-pixels and temporal propagation. In *Scandinavian Conference on Image Analysis*, pages 250–261. Springer.

[Mnih and Hinton, 2010] Mnih, V. and Hinton, G. E. (2010). Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, pages 210–223. Springer.

[Mnih and Hinton, 2012] Mnih, V. and Hinton, G. E. (2012). Learning to label aerial images from noisy data. In *The International Conference on Machine Learning*, pages 567–574.

[Monti et al., 2017] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124.

[Morgan and Bourlard, 1990] Morgan, N. and Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in Neural Information Processing Systems*, pages 630–637.

[Nah et al., 2017] Nah, S., Hyun Kim, T., and Mu Lee, K. (2017). Deep multi-scale convolutional neural network for dynamic scene deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3883–3891.

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1520–1528.

[Park et al., 2019] Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019). Semantic image synthesis with spatially-adaptive normalization. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

[Perazzi et al., 2016] Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732.

[Pinheiro and Collobert, 2014] Pinheiro, P. and Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning*, pages 82–90. PMLR.

[Porter and Duff, 1984] Porter, T. and Duff, T. (1984). Compositing digital images. In *The 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 253–259.

[Ren and Malik, 2003] Ren and Malik (2003). Learning a classification model for segmentation. In *International Conference on Computer Vision*, pages 10–17 vol.1.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

[Rother et al., 2004] Rother, C., Kolmogorov, V., and Blake, A. (2004). " grabcut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314.

[Saeedan et al., 2018] Saeedan, F., Weber, N., Goesele, M., and Roth, S. (2018). Detail-preserving pooling in deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 9108–9116.

[Saito et al., 2016] Saito, S., Yamashita, T., and Aoki, Y. (2016). Multiple object extraction from aerial imagery with convolutional neural networks. *Electronic Imaging*, 2016(10):1–9.

[Sawicki, 2007] Sawicki, M. (2007). *Filming the fantastic: a guide to visual effect cinematography*. Taylor & Francis.

[Schonfeld et al., 2020] Schonfeld, E., Schiele, B., and Khoreva, A. (2020). A u-net based discriminator for generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 8207–8216.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

[Shotton et al., 2006] Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, pages 1–15. Springer.

[Shotton et al., 2009a] Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009a). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23.

[Shotton et al., 2009b] Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009b). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23.

[Shuman et al., 2013] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98.

[Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision*, pages 746–760. Springer.

[Strang, 2006] Strang, G. (2006). *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA.

[Stutz et al., 2018] Stutz, D., Hermans, A., and Leibe, B. (2018). Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27.

[Sun et al., 2018] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943.

[Sun et al., 2004] Sun, J., Jia, J., Tang, C.-K., and Shum, H.-Y. (2004). Poisson matting. In *ACM SIGGRAPH 2004 Papers*, pages 315–321.

[Suzuki, 2020] Suzuki, T. (2020). Superpixel segmentation via convolutional neural networks with regularized information maximization. In *International Conference on Acoustics, Speech and Signal Processing*, pages 2573–2577. IEEE.

[Suzuki, 2021] Suzuki, T. (2021). Implicit integration of superpixel segmentation into fully convolutional networks. *arXiv preprint arXiv:2103.03435*.

[Suzuki et al., 2018] Suzuki, T., Akizuki, S., Kato, N., and Aoki, Y. (2018). Superpixel convolution for segmentation. In *International Conference on Image Processing*, pages 3249–3253. IEEE.

[Suzuki and Aoki, 2018] Suzuki, T. and Aoki, Y. (2018). Graph convolutional neural networks on superpixels for segmentation. 電子情報通信学会論文誌 *D*, 101(8):1120–1129.

[Suzuki and Aoki, 2020] Suzuki, T. and Aoki, Y. (2020). Unsupervised superpixel segmentation via convolutional neural network. 電子情報通信学会論文誌 *D*, 103(10):702–711.

[Takayama et al., 2016] Takayama, S., Suzuki, T., Aoki, Y., Isobe, S., and Masuda, M. (2016). Tracking people in dense crowds using supervoxels. In *International Conference on Signal-Image Technology & Internet-Based Systems*, pages 532–537. IEEE.

[Tao et al., 2018] Tao, X., Gao, H., Shen, X., Wang, J., and Jia, J. (2018). Scale-recurrent network for deep image deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 8174–8182.

[Tasli et al., 2013] Tasli, H. E., Cigla, C., Gevers, T., and Alatan, A. A. (2013). Super pixel extraction via convexity induced boundary adaptation. In *IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE.

[Torralba et al., 2004] Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–II. IEEE.

[Tu et al., 2018] Tu, W.-C., Liu, M.-Y., Jampani, V., Sun, D., Chien, S.-Y., Yang, M.-H., and Kautz, J. (2018). Learning superpixels with segmentation-aware affinity loss. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 568–576.

[Uijlings et al., 2013] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

[Ulyanov et al., 2016] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. Technical report.

[Ulyanov et al., 2018] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018). Deep image prior. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454.

[Van den Bergh et al., 2012] Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., and Van Gool, L. (2012). SEEDS: Superpixels extracted via energy-driven sampling. In *European Conference on Computer Vision*, pages 13–26. Springer.

[van der Walt et al., 2014] van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Veksler et al., 2010] Veksler, O., Boykov, Y., and Mehrani, P. (2010). Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision*, pages 211–224. Springer.

[Voigtlaender et al., 2019] Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B. (2019). Mots: Multi-object tracking and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 7942–7951.

[Von Luxburg, 2007] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

[Weikersdorfer et al., 2012] Weikersdorfer, D., Gossow, D., and Beetz, M. (2012). Depth-adaptive superpixels. In *The IEEE Conference on Pattern Recognition*, pages 2087–2090. IEEE.

[Wu and He, 2018] Wu, Y. and He, K. (2018). Group normalization. In *European Conference on Computer Vision*, pages 3–19.

[Xu et al., 2018] Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., and Huang, T. (2018). Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*.

[Yang et al., 2020] Yang, F., Sun, Q., Jin, H., and Zhou, Z. (2020). Superpixel segmentation with fully convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 13964–13973.

[Yao et al., 2015] Yao, J., Boben, M., Fidler, S., and Urtasun, R. (2015). Real-time coarse-to-fine topologically preserving segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2947–2955.

[Yeh et al., 2017] Yeh, R. A., Chen, C., Yian Lim, T., Schwing, A. G., Hasegawa-Johnson, M., and Do, M. N. (2017). Semantic image inpainting with deep generative

models. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493.

[Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. Technical report.

[Zhang et al., 2019] Zhang, L., Li, X., Arnab, A., Yang, K., Tong, Y., and Torr, P. H. (2019). Dual graph convolutional network for semantic segmentation. Technical report.

[Zhao et al., 2017] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890.

[Zheng et al., 2015] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision*, pages 1529–1537.

[Zhi et al., 2019] Zhi, S., Bloesch, M., Leutenegger, S., and Davison, A. J. (2019). Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 11776–11785.

[Zhou et al., 2016] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2016). Semantic understanding of scenes through the ade20k dataset. Technical report.

[Zhu et al., 2019] Zhu, X., Hu, H., Lin, S., and Dai, J. (2019). Deformable convnets v2: More deformable, better results. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316.

Appendix A

Graph Laplacian

We provide propositions for the unnormalized and normalized graph Laplacian matrices. In the following, we assume that the graph is an undirected, weighted graph with weight matrix W where $w_{ij} = w_{ji} \geq 0$.

Then, the unnormalized graph Laplacian L satisfies Proposition 1 [Von Luxburg, 2007].

Proposition 1.

- *For every vector,* $f \in \mathbb{R}^N$ *we have*

$$f^\top L f = \frac{1}{2} \sum_{i,j=1}^{N} W_{ij}(f_i - f_j)^2. \tag{A.1}$$

- L *is symmetric and semi-positive definite.*

- *The smallest eigenvalue of* L *is* 0*; the corresponding eigenvector is the constant one vector* $\mathbb{1}$*.*

- L *has* N *non-negative, real-valued eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$*.*

Proof.

- Since the diagonal values of the degree matrix are defined as the sum of the column values of the adjacency matrix, $D_{ii} = \sum_j W_{ij}$,

$$\begin{aligned} f^\top L f = f^\top D f - f^\top W f &= \sum_{i=1}^{N} D_{ii} f_i^2 - \sum_{i,j=1}^{N} f_i f_j W_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{N} D_{ii} f_i^2 - 2 \sum_{i,j=1}^{N} f_i f_j W_{ij} + \sum_{j=1}^{N} D_{jj} f_j^2 \right) \\ &= \frac{1}{2}\left(\sum_{i,j=1}^{N} W_{ij} f_i^2 - 2 \sum_{i,j=1}^{N} f_i f_j W_{ij} + \sum_{j,i=1}^{N} W_{ji} f_j^2 \right) \\ &= \frac{1}{2} \sum_{i,j}^{N} W_{ij} (f_i - f_j)^2. \end{aligned} \tag{A.2}$$

- Since the sum of the symmetric matrices is the symmetric matrix, L is symmetric. Moreover, W is a non-negative matrix, and $f^\top L f$ is also non-negative for all $f \in \mathbb{R}^N$. Therefore, since the quadratic form is non-negative for all $f \in \mathbb{R}^N$, L is semi-positive definite [Strang, 2006].

- From the definition of L and D, $L\mathbb{1} = 0$, and since L is semi-positive definite, the smallest eigenvalue of L is 0.

- $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ is a direct consequence of the above propositions.

□

Also, the normalized graph Laplacian L_{norm} satisfies following Proposition [Von Luxburg, 2007].

Proposition 2

- *For every $f \in \mathbb{R}^N$, we have*

$$f^\top L_{\text{norm}} f = \frac{1}{2} \sum_{i,j=1}^{N} W_{ij} \left(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right). \tag{A.3}$$

- *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{norm} with eigenvector $w = D^{1/2}u$.*
- *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem $Lu = \lambda Du$.*
- *0 is an eigenvalue of L_{rw} with the constant one vector $\mathbb{1}$ as an eigenvector; 0 is an eigenvalue of L_{norm} with eigenvector $D^{1/2}\mathbb{1}$.*
- *L_{norm} and L_{rw} are positive semi-definite and have N non-negative real-valued eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_N$.*

Proof.

- It can be proved similarly to Proposition 1 as follows:

$$\begin{aligned} f^\top Lf &= f^\top If - f^\top D^{-1/2} W D^{-1/2} f \\ &= \sum_{i=1} f_i^2 - \sum_{i,j=1} W_{ij} \frac{f_i}{\sqrt{D_{ii}}} \frac{f_j}{\sqrt{D_{jj}}} \\ &= \frac{1}{2} \left(\sum_{i=1} f_i^2 - 2 \sum_{i,j=1} W_{ij} \frac{f_i}{\sqrt{D_{ii}}} \frac{f_j}{\sqrt{D_{jj}}} + \sum_{j=1} f_j^2 \right) \\ &= \frac{1}{2} \left(\sum_{i,j=1} \frac{W_{ij}}{D_{ii}} f_i^2 - 2 \sum_{i,j=1} W_{ij} \frac{f_i}{\sqrt{D_{ii}}} \frac{f_j}{\sqrt{D_{jj}}} + \sum_{j,i=1} \frac{W_{ji}}{D_{jj}} f_j^2 \right) \\ &= \frac{1}{2} \sum_{i,j}^{N} W_{ij} \left(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right)^2 . \end{aligned} \tag{A.4}$$

- From the eigenvalue equation $L_{\text{norm}} w = \lambda w$,

$$\begin{aligned} D^{-1/2} L D^{-1/2} w &= \lambda w, \\ D^{-1} L D^{-1/2} w &= D^{-1/2} \lambda w, \\ L_{\text{rw}} D^{-1/2} w &= \lambda D^{-1/2} w. \end{aligned}$$

  Therefore, $u = D^{-1/2}w$ is derived, and by multiplying it by $D^{1/2}$ from the left, $w = D^{1/2}u$ is derived.

- The proposition is given by multiplying the eigenvalue equation, $L_{\mathrm{rw}}u = \lambda u$, by D from the left.

- From $L\mathbb{1} = 0$, $L_{\mathrm{rw}}\mathbb{1} = D^{-1}L\mathbb{1} = 0$, and then $L_{\mathrm{norm}}D^{1/2}\mathbb{1} = 0$ is derived from $L_{\mathrm{rw}}\mathbb{1} = 0$ and the above propositions.

- L_{norm} is semi-positive definite since its quadratic form is non-negative, and then L_{rw} is also semi-positive definite since $w = D^{1/2}u$. Moreover, L_{norm} and L_{rw} have 0 eigenvalues. Thus, $0 = \lambda_1 \leq \cdots \leq \lambda_N$ is derived.

□

Appendix B

Normalization Layers for Neural Networks

B.1 Batch Normalization

Training neural networks is complicated by the fact that the distribution of each layer ' s inputs changes during training. This fact is sometimes referred to as an *internal covariant shift*. The first work tackled this problem in the scenario of the deep neural networks is batch normalization [Ioffe and Szegedy, 2015]. Batch normalization normalizes the batched inputs as follows:

$$\hat{x}^{(c)} = \frac{x^{(c)} - \mu\left(x^{(c)}\right)}{\sqrt{\sigma\left(x^{(c)}\right)^2 + \epsilon}}, \tag{B.1}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and the standard deviation, and ϵ is a hyperparameter to avoid division by zero. Note that we assume that $x^{(c)} \in \mathbb{R}^{B\times H\times W}$ is the image, where B, H, and W denote the batch size, the image height, and the image width. When training time, $\mu(\cdot)$ and $\sigma(x)$ are computed as follows:

$$\mu\left(x^{(c)}\right) := \frac{1}{BHW}\sum_{b,h,w} x^{(c)}_{b,h,w}, \tag{B.2}$$

$$\sigma\left(x^{(c)}\right)^2 := \frac{1}{BHW}\sum_{b,h,w}\left(x^{(c)}_{b,h,w} - \mu\left(x^{(c)}\right)\right)^2, \tag{B.3}$$

where $x^{(c)} \in \mathbb{R}^{B \times H \times W}$ denotes the c-th channel of the inputs. Typically, after the normalization, the normalized input is transformed by trainable affine transformation as follows:

$$\tilde{x}^{(c)} = \alpha^{(c)} \hat{x}^{(c)} + \beta^{(c)}, \tag{B.4}$$

where $\alpha^{(c)} \in \mathbb{R}$, and $\beta^{(c)} \in \mathbb{R}$ denote trainable parameters.

In inference time, a single input is fed into the model; hence, $\mu\left(x^{(c)}\right)$ and $\sigma\left(x^{(x)}\right)$ cannot be computed. Therefore, batch normalization computes the moving averages of mean and variance and uses them instead.

Batch normalization allows us to use much higher learning rates and to be less careful about initialization. Moreover, because of them, the model requires much fewer training steps compared to models that do not adopt the batch normalization. However, it has a limitation: when $x^{(c)}$ consists of a small number of batches, the batch normalization does not give the model any improvement. Therefore, the normalization methods not depending on batch size are widely studied [Ulyanov et al., 2016, Wu and He, 2018, Ba et al., 2016]. In the next section, we describe instance normalization [Ulyanov et al., 2016], which is used in our work.

B.2 Instance Normalization

Instance normalization [Ulyanov et al., 2016] is the same as batch normalization except that it normalizes the input per sample. The mean and the variance for instance normalization are computed as follows:

$$\mu_{\mathrm{IN}}\left(x_b^{(c)}\right) := \frac{1}{HW} \sum_{h,w} x_{b,h,w}^{(c)}, \tag{B.5}$$

$$\sigma_{\mathrm{IN}}\left(x_b^{(c)}\right)^2 := \frac{1}{HW} \sum_{h,w} \left(x_{b,h,w}^{(c)} - \mu_{\mathrm{IN}}\left(x_b^{(c)}\right)\right)^2, \tag{B.6}$$

where $x_b^{(c)} \in \mathbb{R}^{H \times W}$ denotes the c-th channel of the b-th batch in the inputs. The computed parameters are used to normalize the inputs with eq. (B.4).

B.3 Layer Normalization

Although batch normalization and instance normalization compute the mean and variance over spatial dimension, layer normalization [Ba et al., 2016] includes channel dimension to compute them.

The mean and the variance for layer normalization are computed as follows:

$$\mu_{\mathrm{LN}}\left(x_b\right) := \frac{1}{CHW}\sum_{h,w,c} x_{b,h,w}^{(c)}, \tag{B.7}$$

$$\sigma_{\mathrm{LN}}\left(x_b\right)^2 := \frac{1}{CHW}\sum_{h,w,c}\left(x_{b,h,w,c}^{(c)} - \mu_{\mathrm{LN}}\left(x_b\right)\right)^2. \tag{B.8}$$

Each notation is the same as eq. (B.5). The computed parameters are used to normalize the inputs as follows:

$$\hat{x}_b^{(c)} = \frac{x_b^{(c)} - \mu_{LN}\left(x_b\right)}{\sqrt{\sigma_{LN}\left(x_b\right) + \epsilon}}. \tag{B.9}$$

B.4 Group Normalization

Instance normalization and layer normalization are effective for training sequential models, such as RNN, LSTM, and transformer [Vaswani et al., 2017], and generative models, such as GANs [Goodfellow et al., 2014]. However, these normalization layers have limited success in visual recognition.

Group normalization [Wu and He, 2018] is proposed for computer vision tasks, such as object detection and image segmentation, and produce better results than other normalization layers, especially, when the batch size is small. Group normalization divides channels into some groups and computes the mean and variance for each group in the same way as layer normalization.

Let $G = g_1, \ldots, g_N$ be a set of the groups. The mean and the variance are computed as follows:

$$\mu_{GN}\left(x_b^{(g_n)}\right)_{g_n} := \frac{1}{HW|g_n|}\sum_{h,w,c\in g_n} x_{b,h,w}^{(c)}, \tag{B.10}$$

$$\sigma\left(x_b^{(g_n)}\right) := \frac{1}{HW|g_n|}\sum_{h,w,c\in g_n}\left(x_{b,h,w}^{(c)} - \mu_{GN}\left(x_b^{(g_n)}\right)\right), \tag{B.11}$$

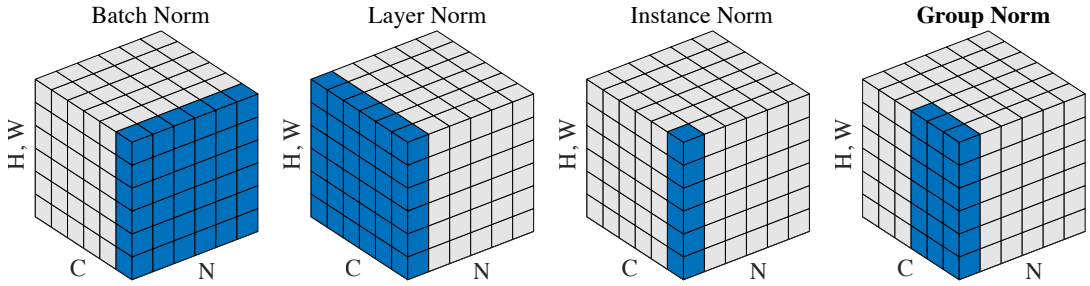


Figure B.1: Illustration of various normalization layers. N, (H, W), and C denote the batch axis, the spatial axes, and the channel axis, respectively. The figure is taken from [Wu and He, 2018].

where $x_b^{(g_n)}$ denotes the g_n group of the b-th batch in the inputs, and $|g_n|$ is the group size. The inputs are normalized as follows:

$$\hat{x}_b^{(c)} = \frac{x_b^{(c)} - \mu_{GN}\left(x_b^{(g_n)}\right)}{\sqrt{\sigma_{GN}\left(x_b^{(g_n}\right) + \epsilon}}, \ s.t.\ c \in g_n. \tag{B.12}$$

Finally, we illustrate all normalization layers in Figure B.1.