Waseda University Master Thesis

Cascaded Fully Convolutional Networks for Object Boundary Detection

44161542-3: YUAN JIANG

Master (Engineering)

Professor Jinglu HU, Supervisor

Neurocomputing System
Information Architecture
The Graduate School of Information, Production and Systems

February 2018

ABSTRACT

In computer vision, object boundary is defined as the enclosed set of edge pixels by which an object is tightly surrounded. Given the object boundary, the shapes and locations of the objects in an image can be easily recognized. However, object boundary detection is still a challenging problem due to the multi-scale object problem, the interference caused by the local edges with less semantic information, and the large variance of backgrounds and scenes. In this thesis, we develop a cascaded framework to improve the performance of the present edge detection algorithms in object boundary detection. Within our framework, present networks can be cascaded one-by-one and trained end-to-end to gain more semantic information from the inputs. We test our method in two scenes: neuronal boundary detection in Electron Microscopy (*EM*) images, and object boundary detection in natural images. Massive experiments and analyses show that the proposed cascaded fully convolutional networks can exactly outperform the competitors with the help of the cascaded structure.

Keywords: Fully Convolutional Networks, Object Boundary Detection, Neuronal Segmentation, Cascaded Structures

ACKNOWLEDGMENTS

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Object boundary detection aims at finding the enclosed boundary in which contains an object. Comparing to traditional edge detection, object boundary detection only considers the boundaries related to objects, despite of the low-level edges which have less semantic information. So the object boundary can be easily applied for instance-level object segmentation[**?**], object proposal[**?**, **?**], biomedical engineering[**?**, **?**, **?**], etc. While it is important and useful in various vision tasks, there are few literature about the object boundary detection due to three challenges: First, object boundary detection is a multi-scale problem; Second, amount of local edges with less semantic information must be inhibited to gain a high-level boundary map; And the large variance of backgrounds and scenes.

Traditional methods mostly focus on extract single-scale features for edge detection[**?**, **?**, **?**, **?**, **?**]. However, objects has various scales in many applications(figure), resulting in the demand of multi-scale descriptors to detect object boundaries. HED[**?**] is the first to adopt the supervised multi-scale feature learning to edge detection, where the feature map from each stage of the convolution neural networks is supervised by the ground-truth labels. We follow the multi-scale feature

extracting proposed by HED[**?**] and extend it to object boundary detection, in which the multi-scale problem is considerably serious.

What's more, the target of traditional edge detection is to identify the discontinuities of image brightness[**?**], regardless of the semantic information which indicate whether the edges enclose an object in it. This is the root why some traditional edge detection methods fail to detect the object boundary. We propose the cascaded fully convolutional networks *(FCNs)*[**?**] with holistic training and testing to overcome the shortages of traditional edge detectors. By recursively connect the FCNs rather than simply stacked the convolution layers, our networks have the ability to inhibit the meaningless edges inside the objects and can be easily interpreted.

Another issue for object boundary detection is the large variance of backgrounds and scenes. Our networks draw inspiration from the related works about neuronal boundary detection in Electron Microscopy *(EM)* images, so we firstly make an effort to apply them on two mainstream datasets for neuronal boundary detection and neuronal segmentation in EM images. To extend the cascaded framework into common cases, we then test our networks in a large natural image dataset, PASCAL Contour[**?**].

We highlight three points in the thesis. (1) extending the multi-scale holistic network of HED[**?**] to the multi-scale multi-recursive network for object boundary detection. (2) cascaded network structure with end-to-end training, which is easy to be integrated with other fully convolutional networks. (3) and systematic experiments in two applications to show the reliability and interpretability of our architecture.

1.2 Application Scenes

1.2.1 EM Images

With the development of Electron Microscopy *(EM)*, neuroscientists and cognitive scientists are able to study the connection between structures and functions of neurons from the high-resolution

EM images[**?**]. The first step for the study is often to reconstruct the structure of neuronal circuits. However it would take thousands of hours for an expert to annotate a large number of EM images, which motivates researchers to explore some automatic reconstructing algorithms. To reconstruct the neuronal circuits, we firstly reconstruct the 2D structure of neurons from a serials of EM images produced by a serial section EM. Then the stack of 2D neuron structure will be concatenated into the 3D neuron structure.

As the basic step, 2D neuron reconstruction has drawn a lot of attentions, especially those adopting the powerful deep neural networks[**?**, **?**, **?**, **?**]. All these approaches convert the reconstruction problem into a 2D boundary detection issue, since the later is easier to be modeled as a binary classification problem and thus there are many literatures and tools can be used to solve it. So long as the neuron boundary is decided, the neuronal circuit can be segmented using simple graph-based methods such as the watershed algorithm[**?**].

Neuronal boundary detection in EM images is a special case of object boundary detection. There are several commons between neuronal boundary detection and object boundary detection, for example, both them focus on the high-level boundary, with less consideration about the low-level local edges. In neuronal boundary detection, to be specific, the low-level local edge includes the ones from background and EM noises, and also the ones produced by confounding structures like nucleus, mitochondria, etc. Hence, traditional edge detectors[**?**, **?**, **?**, **?**] often fail to detect the boundary in EM images (figure). Deep neural networks are also widely used to detect neuronal boundary and have gained promising progress[**?**, **?**, **?**], while lack of interpretation. Different from all these works, we use a lot of experiments not only to show the performance of the proposed cascaded fully convolutional networks in neuronal boundary detection, but also to explain how the proposed architecture removes the insider local edges and background noises, and strengthens the boundary with low-contrast.

We compare our method with competitors on two mainstream datasets for neuronal segmentation, the Piriform dataset[**?**] and the ISBI 2012 Challenge dataset[**?**].

1.2.2 Natural Images

Although our idea is firstly motivated by the problem we met in neuronal boundary detection, we also take lots of attentions in extending the proposed models into more commonly used applications, such as the object boundary detection in natural images. Given the object boundary map of a natural image, we can easily obtain the object proposal, object segmentation[**?**, **?**], object detection[**?**, **?**], and recognize object using shape based classification methods[**?**, **?**, **?**].

Object boundary detection in natural images is still a difficult problem due to the fact that objects vary in colors, brightness, scales, shapes, textures and gestures. One can hardly recognize the boundaries of all kinds of objects in natural images with only one of the features. Thus, many traditional edge detection methods[**?**, **?**, **?**] do not perform well when directly applying them for high-level object boundary detection.

Performance benchmark is done on the famous PASCAL VOC dataset[**?**]. PASCAL VOC is one of the most famous object detection and segmentation challenge held once a year. We collect the training and validation data from PASCAL VOC 2007, to PASCAL VOC 2012. Ground-truth boundary maps are obtained from the instance-level segmentation annotations provided by the official. Comparison results show our cascaded network outperforms not only the traditional edge detectors but also the deep networks without cascade.

1.3 Organization of the thesis

The rest of this thesis is structured as follows: Chapter II introduces some related works in traditional edge detection methods and neuronal boundary detection methods, concluding their progresses and shortages. In Chapter III, we describe the proposed cascaded fully convolutional networks, giving an insight into why we design such a architecture and how it works. To evaluate the performance and verify the interpretation, Chapter IV tests our models on three datasets, two for neuronal boundary detection / segmentation in EM images, and one for object boundary detection

in natural images. The thesis ends in Chapter V with conclusions on contributions and future works of this research.

Chapter 2

Related Work

Comparing with traditional edge detection, object boundary detection focuses on the high-level semantic information about the objects appeared in the image. It is useful in many applications and has many advantages to the local edges, however, object boundary still lacks of researches in literature. So we will start from traditional edge detection methods to seek the recommendation. On the other hand, neuronal boundary detection in EM images has achieved considerable progresses. Deep neural networks perform well in the neuronal boundary detection, inspiring us to explore a more powerful and explainable one to improve the boundary detection results for not only the EM images, but also the natural images.

2.1 Traditional edge detection

Edge detection is a fundamental task with a long history in computer vision. Contrast to object boundary detection, local edge detection concerns all the pixels with color, brightness, texture, and other properties changing sharply in local parts of an image.

2.1.1 Hand-crafted feature based methods

Canny[**?**] can be seen as the pioneer of computationally extracting edges from digit images. Canny uses image gradient to get the edges. Firstly, the input RGB image should be converted to a greyscale one, whose 2D gradient map will be next calculated using the Sobel[**?**] filters. Then the non-maximum suppression is applied in the gradient map to get the thinned edges. After that Canny proposed to adopt an empirically selected double threshold to filtering the edges, where the edges higher than the high threshold will be marked as the "strong edges", the edges lower than the low threshold will be marked as background, and the rest of edges are so called the "weak edges". Canny descriptor use only the brightness gradient of the image, but has almost no ability to utilize the color and texture information of the image, much less the semantic information.

The followers [**?**, **?**, **?**, **?**] tend to combine brightness, color and texture gradients to make a better use of RGB inputs. All these methods are patch-based learning methods. In training phase, random sampled mini patches are cropped from original input image. Features of these patches are extracted using various cues mentioned above. Then the features of training samples will be fed into a two-class classifier, such as Support Vector Machine and Random Forest, to judge whether the central pixel of the sample is an edge pixel or not. In test phase, a one-step sliding window[**?**] will be applied for a dense probability map about edge. To detect edges with more semantic meaning, the *gPb*[**?**] computes gradients at different scales, but resulting in a huge increase of computational load. Lim et al.[**?**] rather propose a structured feature learning pipeline for an effective and fast detection.

All these methods choose hand-crafted features based on local cues, with few design for object-level boundary detection. Experiments in Chapter IV will show their shortages in object boundary detection tasks.

2.1.2 Deep learning based methods

Deep Learning, especially the Convolutional Neural Networks *(CNNs)*, developed rapidly in recent years. The CNNs showed their power firstly in the large-scale object recognition challenges. AlexNet, proposed by Krizhevsky et al.[**?**] in 2012, improved the top-5 rate in ILSVRC-2010 and ILSVRC-2012 competition by around 8%, which drives researchers' trying on applying the deep feature extraction for most of the computer vision tasks. VGGNet[**?**], GoogLeNet[**?**] and ResNet[**?**] are the followers of [**?**]. All these works tend to find the ways to train a deeper or wider network so that the abstract object-level feature can be better learned and extracted. In the next years, CNNs are also widely applied for object detection[**?**], object segmentation[**?**], and object contour detection[**?**, **?**, **?**].

Similar to [**?**], Shen et al.[**?**] cluster the edge structures into several subclasses and convert the two-class classification task to a multi-class one. By this way, the deep networks are able to learn different model parameters for each subclass, and the results are more interpretable. [**?**] utilizes the deep features to capture high-level object information and outperforms all the traditional hand-crafted feature based methods. However, in testing phase, the features extracted by fully connected layer must be fed into a structured random forest classifier[**?**] for the final contour detection result. The two-step learning limits the performance of deep networks. What's more, due to the existing fully connected layers, inputing the original images with various scales is forbidden. So the authors[**?**] have to adopt the patch-based training and testing, leading to a low detection speed.

In 2014, Fully Convolutional Networks *(FCNs)*[**?**] are proposed to handle with the per-pixel annotation problem, such as the object segmentation in the famous PASCAL VOC Challenge[**?**, **?**]. Developed from VGGNet, FCNs remove the last fully connected layers of it and replace with 1×1 convolution layers, which releases the limitation of the input size. A non-parameter deconvolution layer is followed by the last convolution layer to up-sample the low-resolution feature maps into probability maps with same resolution of the input. Finally, after properly cropping, the output probability maps will have the same size with the input. When training, these probability maps

can be used to calculate the pixel-wise loss with ground-truth labels. FCNs enable the end-to-end training and testing of CNNs for such a pixel-wise prediction task. With FCNs, the precision and speed will not be restricted by the sliding window scheme.

Features from different stages of the convolution neural networks have the multi-scale information[**?**]. Xie et al.[**?**] propose to make a full use of these information by extracting features from each stage of the FCNs, then up-sampling them to the same size with the input, which results in multi-scale outputs called side-outputs. All the side-outputs and the fusion of these side-outputs will contribute on the loss, where both the low-level stages with high resolution but little semantic information, and the high-level stages with low resolution but much semantic information can be trained easilier. Benefited from the holistic training and multi-scale features, HED[**?**] achieves great performance in the famous edge detection benchmark, BSDS500[**?**], improving the F-measure[**?**] from 0.756[**?**] to 0.782. Note that the F-measure of human being's annotation is 0.8.

2.2 Object Boundary Detection in Natural Images

Although HED[**?**] and other edge detection methods[**?**, **?**, **?**] have achieved considerable success in relatively low-level edge detection, they perform bad when applied for the high-level *"object-only"* boundary detection[**?**]. Fully Convolutional Encoder-Decoder Network *(CEDN)*[**?**] is the pioneer to distinguish object-level boundary detection from traditional local-level edge detection. CEDN uses VGG-16[**?**] as the encoder and designs an approximately symmetric but light-weighted decoder. During training, it fixes the model parameters of the encoder and only updates the parameters of the decoder.

CEDN outperforms all the competitors on the large PASCAL VOC Contour Dataset[**?**] in object boundary detection benchmark. However, it still has several problems: (1) the encoder network does not benefit from the training data at all; (2) to achieve the performance in the leaderboard, it requires all the training data to be processed for 30 times, due to the comparatively deep architec-

ture.

2.3 Neuronal Boundary Detection in EM Images

Neuronal segmentation is the fundamental step to learn the structures of neural cells and analyze the relationship between the structures and the functions[**?**, **?**, **?**]. In early days, however, it would take an expert with professional knowledge lots of time to recognize the segment of neuronal circuits[**?**] in massive EM images, which motivated researchers to find a automatic algorithm to annotate the neuronal segments. People find the key to the neuronal segmentation is to detect the membranes of the cells, while suppressing the local edges within them[**?**, **?**, **?**, **?**]. So that graph-based algorithms such as watershed[**?**, **?**, **?**] and graph cut[**?**] can be directly used to convert the boundary to segmentation.

To detect neuronal boundary is to extract membranes in an EM image. In this degree, Neuronal Boundary Detection can be seen as a special case of Object Boundary Detection, where the *"object"* in the former is only the neuron circuit.

Deep learning based methods are able to meet the demand of different applications, by learning the parameters of not only the classifiers, but also the feature extractors. With the rapid development of deep networks, adopting them into the hard neuronal segmentation tasks becomes popular among researchers. Comparing with traditional hand-designed features, deep features need less experts' knowledge to design the feature extractors and can make a full use of the data, which is proved to be effective in neuronal boundary detection tasks[**?**, **?**, **?**, **?**, **?**]. As the winner of the famous ISBI 2012 EM Segmentation Challenge[**?**], Ciresan et al.[**?**] successfully apply the deep neural networks with a stack of *convolution-subsampling* units for membrane pixel detection. Ronnerberger et al.[**?**] propose U-net, which reuses the features of the layers in the encoder part by merging them with the layers in the decoder part, to detect neuronal boundary and segment EM circuits. With the benefit of fully convolutional network structure and multi-level feature con-

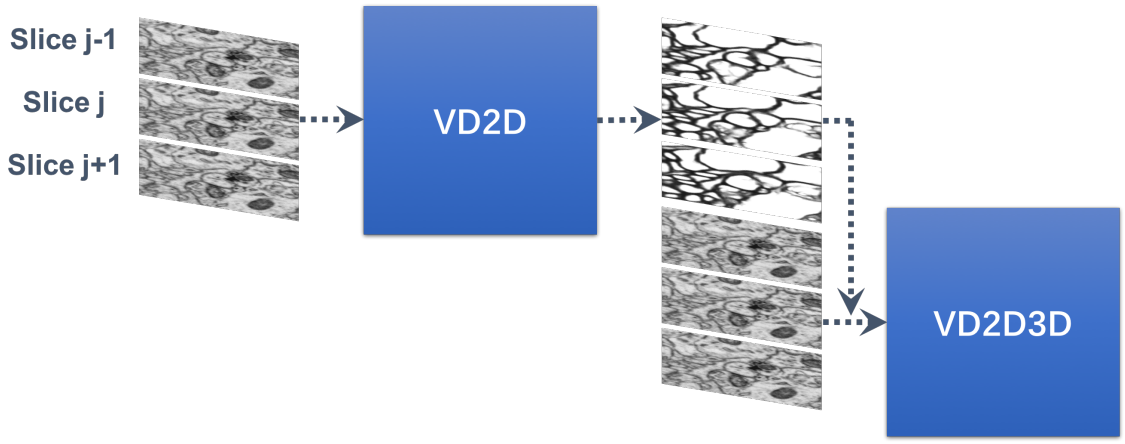

Figure 2.1 Illustration of VD2D3D, a recursive deep network with the stepwise training.

catenating, U-net shows the efficiency in neuronal segmentation by outperforming the competitors in ISBI 2012 by a large margin. With the proposal of ResNet[**?**] in object recognition and detection, Quan et al.[**?**] and Fakhry et al.[**?**] try to implement the deep residual uints in their fully convolutional networks, which is also proved to be effective in neuronal boundary detection issues.

However, most of these works engraft the state-of-the-art network architectures proposed in other vision works as a **black-box** to boost the performance of neuronal boundary detection. It is necessary to seek an interpretable and effective approach, so that it can be theoretically comprehensive and enlighten the relative works in the future.

Most related to our work is Very Deep 2D-3D network *(VD2D3D)* proposed by Lee et al.[**?**]. A Very Deep 2D *(VD2D)* network is firstly trained using patches randomly cropped from the images. The model of VD2D will be then used to produce the predictions of a stack of images produced by the serial section EM. [**?**] believes the adjacent slices will help to detection the boundary, so VD2D3D concatenates the raw image and the prediction of its adjacent slices, produced by VD2D, as the new training materials of VD2D and recursively fine-tunes last several layers of VD2D. We find in experiments that [**?**] has three main problems: (1) it lacks of multi-scale features to better represent the complex neural membranes; (2) stepwise training limits the learning ability of the recursive deep networks; (3) patch-based testing relies on the sliding window pipeline, which

seriously slow down the testing speed. Inspired by [**?**], we develop a cascaded fully convolutional network with multi-recursive inputs and train it end-to-end. The proposed network overcomes the shortages of [**?**] to achieve the state-of-the-art performance on a neuronal segmentation benchmark[**?**] and a large natural object boundary detection benchmark[**?**].

Chapter 3

Methodology

In this part, we will introduce the proposed Cascaded Fully Convolutional Networks for object boundary detection. First, the cascaded network structure with multi-recursive inputs is presented. Then we will describe our end-to-end training, where every part of the cascaded network can benefit from the training data. Next, in Section 3.3, we will demonstrate how to apply the learned fully convolutional model for the accurate yet fast detection. In the last section of this chapter, we spend a lot of ink to lead you into the insight of the proposed architecture, explaining how it works.

3.1 Network Structure

As shown in Fig. 3.1, we considered several methods to assemble our cascaded networks. Fig. 3.1(a) shows the structure of single-recursive-input cascaded fully convolutional networks, which feeds the final output of the former sub-network into the next sub-network, recursively. Each of these sub-networks is the fully convolutional network with the same structure but unshared model parameters. Fig. 3.1(b) illustrates the cascaded structure with multi-recursive inputs, our final choice. In this design, the predictions from not only the last layer but also the side-output layers[**?**] will be the recursive inputs of the next stage, so we name it as multi-recursive-input design. With

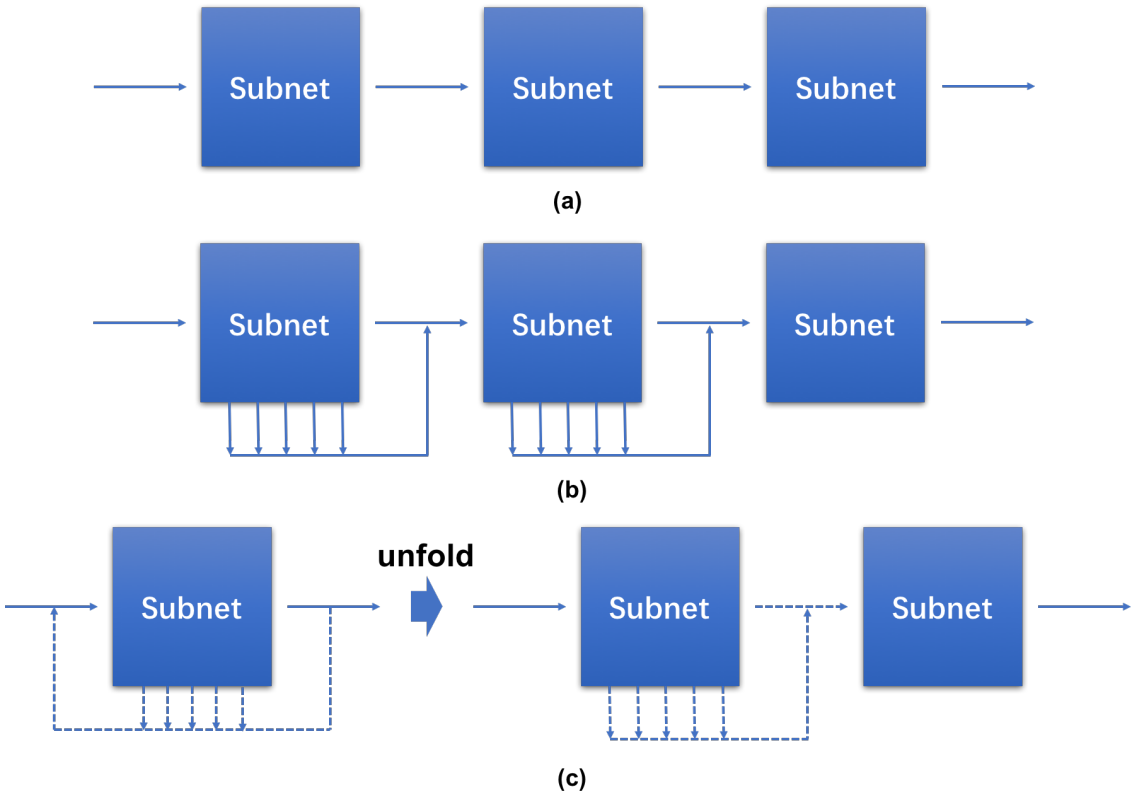

Figure 3.1 Three types of cascade-like networks: (a) cascaded network with single-recursive input; (b) cascaded network with multi-recursive inputs; (c) single-stage network with recursive inputs to fine-tune itself.

contrary to the single-recursive-input one, the proposed architecture makes the full use of features with multiple scales extracted from the former sub-network. Fig. 3.1(c) is another design for constructing such a multi-stage network, where there is only one sub-network, but fine-tuned time after time by recursively feeding the predictions into itself. To unfold the recursive connection of the left side of Fig. 3.1, we can get the equivalent network as shown in the right side. The unfolded version is much similar to the proposed architecture. However, we train all the parameters of sub-networks jointly in the proposed cascaded network, while Fig. 3.1(c) fixes the parameters of former sub-networks and only fine-tune the ones of the last sub-networks. Experiments in Chapter 4 show our design is more effective and efficient than the others.

To be more specific, our network used in object boundary detection can be illustrated as Fig.

3.2. It consists of several stages, three in the default setting, each of which is a fully convolutional network. By default, each sub-network is built based on the holistic-nested network[**?**], which is converted from the famous VGG-16[**?**] network. So the subnet has five stages of the convolution-pooling structure, of which the receptive fields are increased with the stepwise dilated strides. For each level, the side-output is obtained by up-sampling the feature from the last convolution layer of the level and activating it using a sigmoid layer. We mark the side-outputs of level 1 to 5 as $S_1^{(p)}$, $S_2^{(p)}$, $S_3^{(p)}$, $S_4^{(p)}$, $S_5^{(p)}$, respectively, where $p \in 1,2,3$ denotes the order of the sub-network.

Benefited from the receptive field sizes increasing, side-outputs are able to capture features in different scales naturally, without input scaling. Multi-scale features are then fused using the 1×1 convolution filters to produce the final predictions for the stage 1, 2, 3, named $F^{(1)}$, $F^{(2)}$, $F^{(3)}$, respectively. And the best prediction we used in benchmarks is the fusion of $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$. As mentioned above, there are abundant multi-scale information stored in the side-outputs, so we concatenate all the side-outputs with fused prediction and the raw image along with the color channel as the input of the next stage, which called the multi-recursive-input schema.

We choose HED[**?**] as the sub-network in each stage to carry on the multi-scale feature extraction and holistic deep supervision proposed by [**?**]. Here, the deep supervision refers to use the ground-truth to supervise the prediction of not only the last output, but also the side-outputs[**?**], which avoids gradient vanishing in some degree and allows shallow layers efficiently learn the parameters from data.

3.2 Training Phase

Object Boundary Detection can be formulated as a pixel-wise two-class classification problem. Given a training image $X = (X(j), j = 1,2,...,|X|)$ with totally |X| pixels over the spatial dimensions, our goal is to classify each $\tilde{Y}(j)$, the label of pixel $X(j)$, as a boundary pixel ($\tilde{Y}(j) = 1$) or not ($\tilde{Y}(j) = 0$). And we use $\tilde{Y} = (\tilde{Y}(j), j = 1,2,...,|X|)$ to denote the corresponding detected boundary

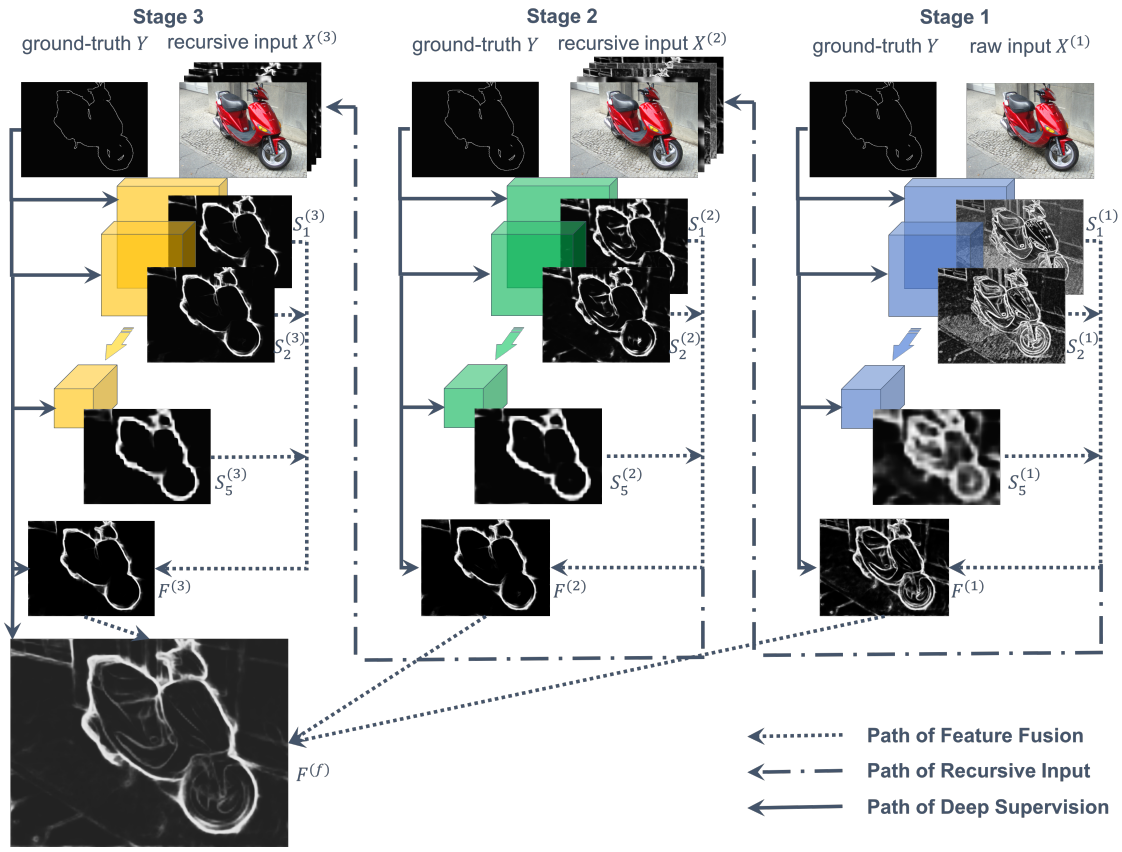

Figure 3.2 Illustration of the proposed cascaded fully convolutional network for object boundary detection.

map. Considering the mini-batch with size equal to Z, and the network with P sub-networks and Q levels in each sub-network, we demonstrate our end-to-end training as follows.

3.2.1 Formulation of the Multi-recursive-input

Let $S_p^{(q)}$ be the side-output of the p^{th} level in the q^{th} sub-network, $F^{(q)}$ and $X^{(q)}$ be the fused prediction and training input in the q^{th} sub-network, now we have:

$$X^{(q)} = \begin{cases} X & ,q = 1 \\ X \circ S_{q-1}^{1} \circ S_{q-1}^{2} \circ \cdots \circ S_{q-1}^{Q} & ,q > 1 \end{cases} \tag{3.1}$$

Here $\circ$ represents the concatenating operation along with the color channel.

3.2.2 Loss Function

To demonstrate the loss function of our cascaded network, we firstly present the loss function of side-output $S_p^{(q)}$:

$$\begin{aligned} l_p^{(q)}(\mathbf{W}^{(q)}, \mathbf{w}_p^{(q)}; X^{(q)}, Y) = & -\alpha \sum_{j \in |B|} log(1 - sigmoid(s_p^{(q)}(j))) \\ & -(1-\alpha) \sum_{j \in |\bar{B}|} log(sigmoid(s_p^{(q)}(j))) \end{aligned} \tag{3.2}$$

where $\mathbf{W}^{(q)}$, $\mathbf{w}_p^{(q)}$, $|B|$ and $|\bar{B}|$ refer to the model parameters of the q^{th} sub-network, ones of the $S_p^{(q)}$, the sets of boundary and non-boundary ground-truth annotations, respectively. α is the ratio of $|\bar{B}|$ and $|B|$, which cancels out the huge imbalance of positive samples (boundary pixels) and negative samples (non-boundary pixels) in some degree[**?**].

Let $\beta_p^{(q)}$ be the weight of $S_p^{(q)}$. Given the notation as follows:

$$\begin{aligned} \mathbf{W} &= \{\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}^{(q)}\}, \\ \mathbf{w} &= (\mathbf{w}_p^{(q)}, p from 1 to P and q from 1 to Q), \end{aligned}$$

the loss of all the side-outputs in the q^{th} sub-network, L_S, is the sum of the losses of all its side-outputs and the fusion output.

$$L_S(\mathbf{W}, \mathbf{w}; X, Y) = \sum_{q=1}^{Q} \sum_{p=1}^{P} \beta_p^{(q)} l_p^{(q)}(\mathbf{W}^{(q)}, \mathbf{w}_p^{(q)}; X^{(q)}, Y) \tag{3.3}$$

All the side-outputs in one stage are fused through a 1×1 convolution layer as mentioned in Section 3.1. The fusion of q^{th} sub-network is defined as:

$$F^{(q)} = \sum_{p=1}^{P} \theta_p^{(q)} S_p^{(q)} \tag{3.4}$$

where $\theta_p^{(q)}$ is the fusion weight of $S_p^{(q)}$.

The fusion loss of q^{th} stage is:

$$\begin{aligned} l_f^{(q)}(\mathbf{W}^{(q)}, \mathbf{w}^{(q)}; X^{(q)}, Y) = &- \alpha \sum_{j \in |B|} log(1 - sigmoid(F^{(q)}(j))) \\ &- (1-\alpha) \sum_{j \in |\bar{B}|} log(sigmoid(F^{(q)}(j))) \end{aligned} \tag{3.5}$$

where $\mathbf{w}^{(q)}$ is the set of the parameters for all the side-outputs in stage q.

Similar to Function 3.4, the final prediction is produced by:

$$F^{(f)} = \sum_{q=1}^{Q} \phi^{(q)} F^{(q)} \tag{3.6}$$

where $\phi^{(q)}$ is the fusion weight of $F^{(q)}$.

The loss of all the fusion outputs is:

$$L_f(\mathbf{W}, \mathbf{w}, \theta, \phi; X, Y) = \sum_{q=1}^{Q} \beta_f^{(q)} l_f^{(q)}(\mathbf{W}^{(q)}, \mathbf{w}^{(q)}; X^{(q)}, Y) \tag{3.7}$$

where $\theta = (\theta_p^{(q)}, q = 1, 2, ..., Q, p = 1, 2, ..., P)$, $\phi = (\phi^{(q)}, q = 1, 2, ..., Q)$ and $\beta_f^{(q)}$ denotes the loss weight for each sub-network.

Finally, we train all the parameters jointly in an end-to-end framework by minimize the following total loss:

$$L_t(\mathbf{W}, \mathbf{w}, \theta, \phi; X, Y) = L_S(\mathbf{W}, \mathbf{w}; X, Y) + L_f(\mathbf{W}, \mathbf{w}, \theta, \phi; X, Y) \tag{3.8}$$

3.3 Testing Phase

When testing, we input an image X into the first sub-network and obtain the fusion of each sub-network as Function 3.4, step by step. Considering the q^{th} stage as a function $f^{(q)}$, the final prediction $\tilde{Y}$ can be produced by:

$$\tilde{S}_p^{(q)} = f^{(q)}(X^{(q)}, (\mathbf{W}^{(1)})^*, ..., (\mathbf{W}^{(Q)})^*, (\mathbf{w}_1^{(Q)})^*, ..., (\mathbf{w}_P^{(Q)})^*) \tag{3.9}$$

$$\tilde{F}^{(q)} = \sum_{p=1}^{P} (\theta_p^{(q)})^* \tilde{S}_p^{(q)} \tag{3.10}$$

$$\tilde{F}^{(f)} = \sum_{q=1}^{Q} (\phi^{(q)})^* \tilde{F}^{(q)} \tag{3.11}$$

$$\tilde{Y} = sigmoid(\tilde{F}^{(f)}) \tag{3.12}$$

3.4 Model Interpretation

3.4.1 Cascaded Architecture *vs.* Single-stage Architecture

Such a cascaded architecture for object-level boundary detection can be interpreted as follows:

On one hand, our eyes can capture various levels of data due to the information re-organization achieved by the multi-stage vision system in our brain.The proposed cascaded architecture mimics the multi-stage vision system of human-beings, by introducing the recursive links to re-filter the multi-scale features produced by the previous stage. As illustrated in Fig. 3.4, to compare the side-outputs and fusions in row (in the same level), we find the boundary map of each stage is *"clearer"* than the former stage. The cascaded network enables the later layers to have a higher-level scope, which breaks the limitation of the receptive filed size existing in a single-stage network. Comparing with a considerably deep single-stage network which has the same receptive size, in other words, our cascaded network will not suffer from the gradient vanishing and be much easier to be trained.

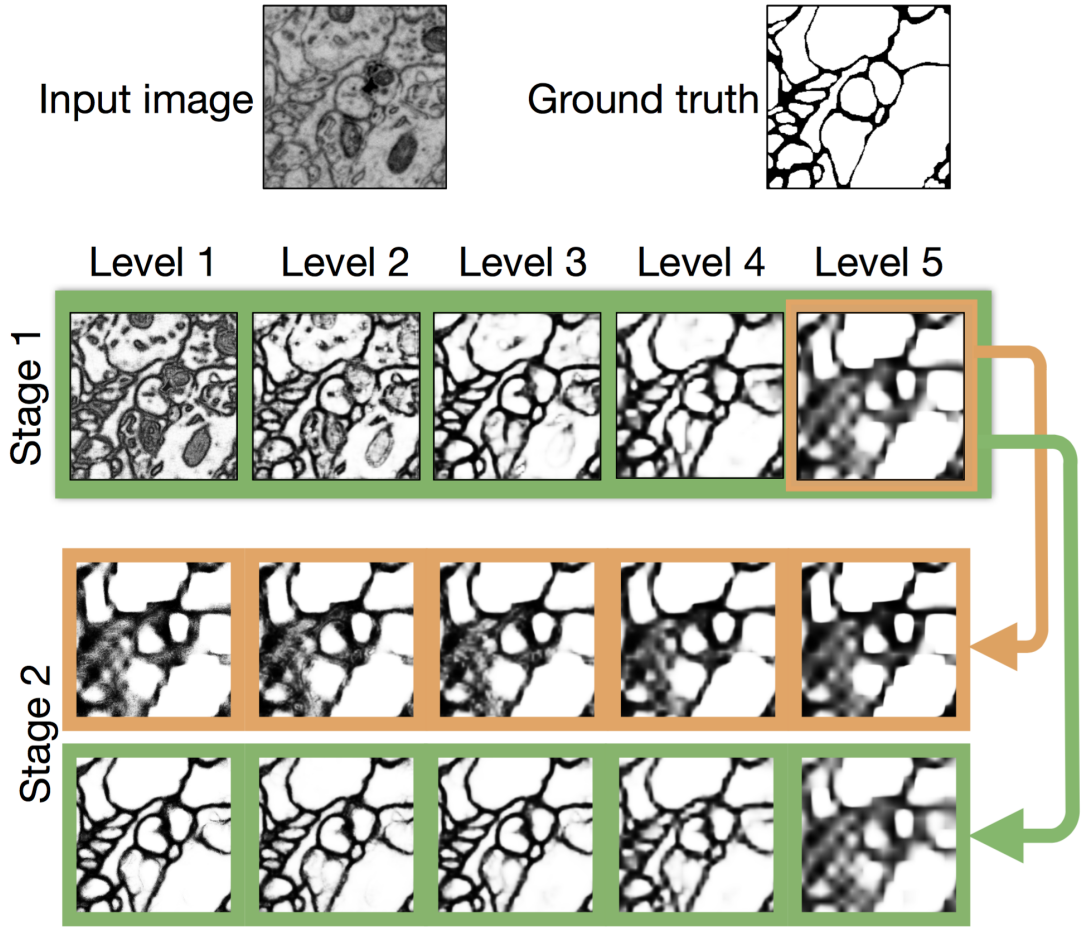

Figure 3.3 Multi-recursive-input *vs.* Single-recursive-input.

On the other hand, human can see both the local details and the object-level abstracts from an image, while not confused by the complex multi-scale information. It is mainly because our brain has a multi-stage vision system to filter these information and select the useful information from the right scale[**?**]. the joint training and feature fusion of the proposed method let data drives the choice from the results of all the stages and levels.

3.4.2 Multi-recursive-input *vs.* Single-recursive-input

One of the biggest differences between our work with others[] is the multi-recursive-input. The qualitative comparison can be found in Fig. 3.3, where the single-recursive-input only reuses the large scale features in the next stage and thus loses the ability to accurately localize the boundary.

3.4.3 End-to-end Training *vs.* Stepwise Self-tuning

To train our network end-to-end, the former stages and the next stages can have an effect on each other. Instead of fixing the model parameters of the previous stages, we let data teach how the network works. With enough training data and the proposed easy-to-train network architecture, our network is able to adjust all the parameters jointly and benefit from it to achieve a better performance than trained stepwise. Refer to Chapter 4 for quantization results and details.

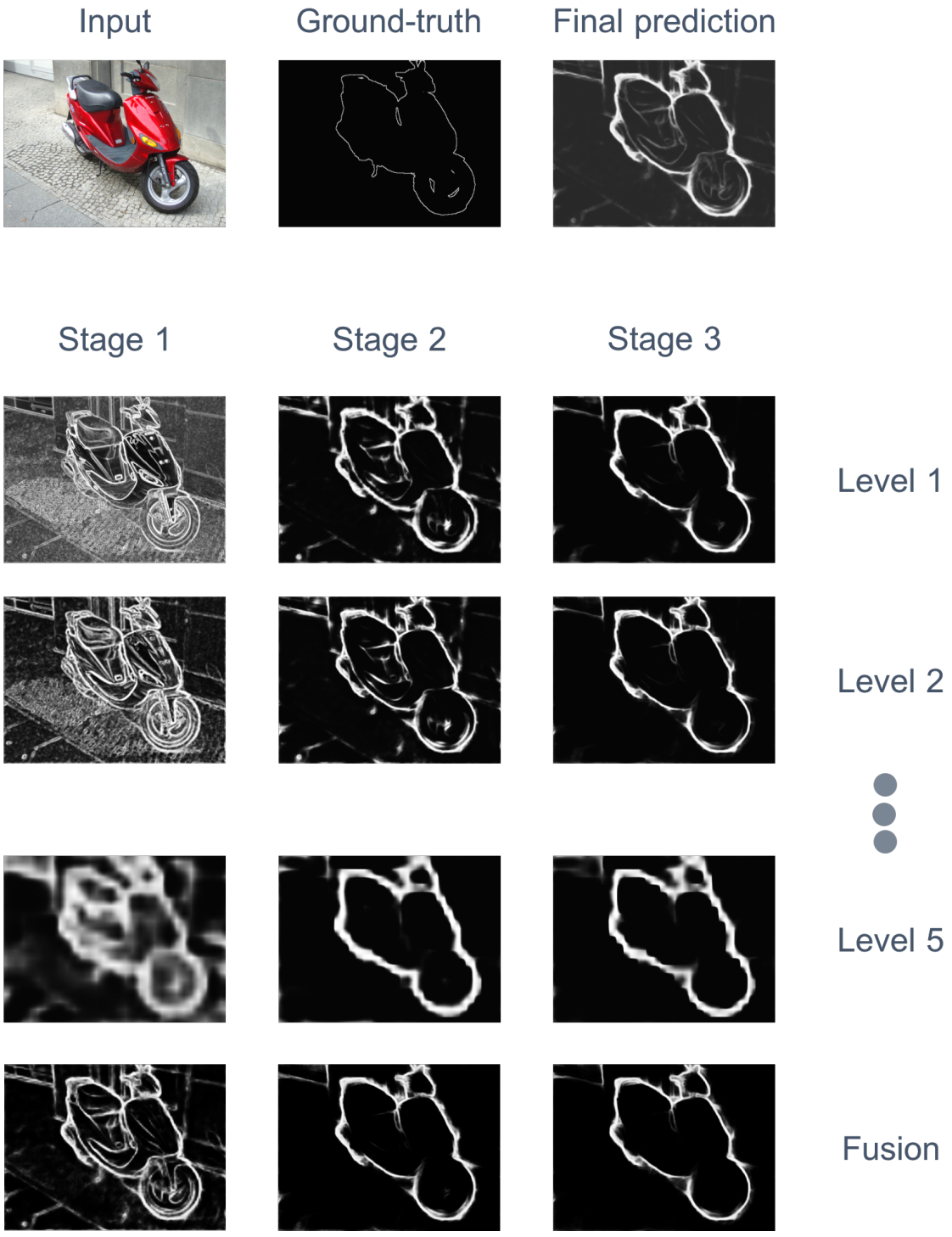

Figure 3.4 Prediction exmaples from different side-outputs and fused outputs in different stages.

Chapter 4

Experiments

As addressed in Chapter 1, we firstly test our cascaded fully convolutional network in the task of neuronal boundary detection on two mainstream datasets, Mouse Piriform Cortex Dataset[**?**] and ISBI 2012 EM Segmentation Dataset[**?**], then extend to adapt with the object boundary detection in natural images on the large object boundary dataset, PASCAL VOC Contour Dataset[**?**]. Besides, we will give the comparative experiments to explain how the performance is achieved by the proposed network structure.

4.1 Neuronal Boundary Detection in EM Images

Neuronal boundary detection in EM images is usually formed as the segmentation annotation problem (Fig. 4.1). In training phase, we compute the 2D gradient in the segmentation ground-truth to obtain the boundary annotation. And then in testing phase, we apply the graph-based algorithms such as watershed algorithm to transfer the boundary prediction into segmentation, since the evaluation metric is based on the segmentation results. In this experiment, we firstly detect the membrane boundary and then apply graph-based watershed[**?**] to it for a segmentation result.

4.1.1 Evaluation Metric

We follow the metric proposed in [**?**], using the Rand F-score to report the performance of the segmentation. As the harmonic average of the Rand merge score and Rand split score, marked as $V^R_{merge}and$ and $V^R_{split}and$ respectively, the Rand F-score:

$$V^{Rand}_{Fscore} = \frac{2V^{Rand}_{merge}V^{Rand}_{split}}{V^{Rand}_{merge} + V^{Rand}_{split}} \tag{4.1}$$

$$\tag{4.2}$$

and

$$V^{Rand}_{merge} = \frac{\sum_{ij} p(i,j)^2}{\sum_i (\sum_j p(i,j))^2} \tag{4.3}$$

$$V^{Rand}_{split} = \frac{\sum_{ij} p(i,j)^2}{\sum_j (\sum_i p(i,j))^2} \tag{4.4}$$

where $p(i,j)$ indicates the number of pixel pairs located in the i^{th} segment of the detected segmentation and the j^{th} segment of the ground-truth segmentation.

Rand split score is high for those results with few split errors, while Rand merge score is high for those results with few merge errors. Generally there will be a trade-off between the Rand split score and the Rand merge score for an image or a dataset. The Rand F-score here is to define the trade-off score, just like the role of the F-score or F-measure in many object detection benchmarks to average the precision metric and recall metric.

The Precision (rand merge)-recall (rand split) curves *(*PR-curves) can be generated by varying the threshold for boundary binarization[**?**].

4.1.2 Mouse Piriform Cortex Dataset

Mouse Piriform Cortex Dataset[**?**] contains 4 stacks of grayscale EM images and their corresponding segmentation annotations(Table. 4.1), where the stack 1 for validation and the stack 2, 3, 4 for training.

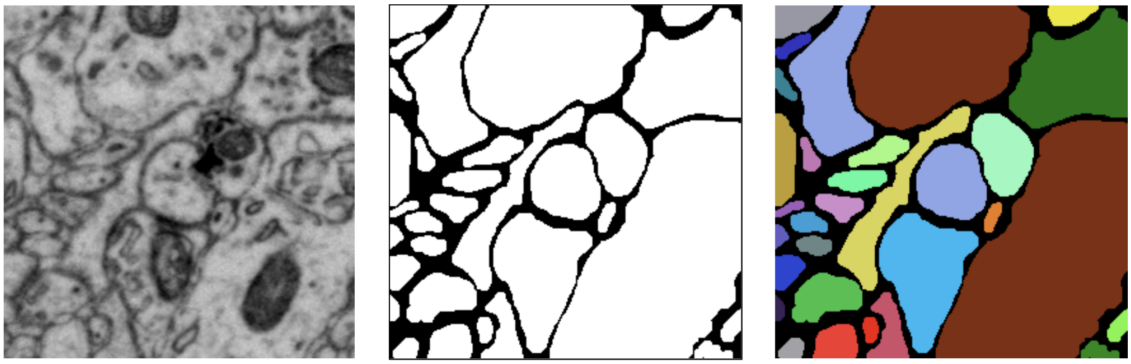

Figure 4.1 (b)Neuronal boundary detection and (c) segmentation can be easily converted from each other, which we used in the experiments to training and testing on Mouse Piriform Cortex Dataset[**?**] and ISBI 2012 EM Segmentation Dataset[**?**]. (1) By applying the graph-based algorithms such as watershed[**?**], we can transfer the boundary prediction into segmentation. (2) By calculating the 2D gradient in the segmentation ground-truth, the boundary annotation can be obtained.

Table 4.1 Mouse Piriform Cortex Dataset

	stack 1	stack 2	stack 3	stack 4
Slices	168	170	169	121
Scales	255*255	512*512	512*512	256*256
Usage	Validation	Training	Training	Training

The annotation of EM images is laborious and time consuming[**?**], leading to the lack of annotated training samples in EM boundary detection. However, the deep networks often require thousands of training data to drive the optimization algorithms. So we augment the raw data to generate sufficient data for training, by rotating the raw image into four directions $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$, then flipping them with three cases (horizontal flipping, vertical flipping, no flipping), and rescaling them to three scales $(0.8, 1.0, 1.2)$, resulting in the training data 36 times the raw images.

To train such a considerably large network from scratch is not necessary nor effective, so we follow the framework proposed by a lot of works[**?**, **?**, **?**] to initialize model parameters of the first stage with the VGG-16 models pre-trained in ImageNet[**?**]. To ease the multi-stage training, we train a single-stage model first. Then use it to initialize the first stage of a two-stage network, leaving the second stage is initialized randomly, by which our multi-stage networks can reuse the relatively low-level filters learned by the single-stage network. Note that the proposed cascaded network is still trained end-to-end because we do not fix any parameters in the final training step but train all the parameters jointly instead.

For experiment settings, we use the SGD algorithm[**?**] to optimize the network, with the momentum of 0.9, the weight decay of 2×10^{-4}, and a stepwise learning rate from 1e-8 to 1e-9(after 20000 iterations). By default, there are 5 convolution levels in each subnet. The training for the takes 20000 iterations for the single-stage network, and 10000 iterations for each of the sequential sub-networks.

Quantitative results are shown in Table 4.2 and Fig. 4.2. With stacking the sub-networks, our method can maintain both a high recall and a high precision. Compared with the current state-of-the-art method VD2D3D[**?**], a deep network with the parameters of several layers fine-tuned by recursively input the prediction into itself, our 2-stage and 3-stage cascaded network can outperform it by around 1% and 1.5% respectively. We argue that such a improvement is meaningful in neuronal boundary detection, since the precise boundary annotation is vital for neural circuits reconstruction and has a low tolerance of errors.The architecture of VD2D3D is much like the Fig. 3.1(c), while the VD2D3D not only feeds the boundary map of the raw input but also generates the predictions of neighbor slices and concatenates all them to enhance the learning (Fig. 2.1). Our networks, taking advantage of the multi-recursive-input and end-to-end training, can outperform the VD2D3D without any additional 3D context information.

Table 4.2 Rand F-scores on Mouse Piriform Cortex Dataset[**?**]

	Rand Split Score	Rand Merge Score	Rand F-score
N4[**?**]	0.9010	0.9619	0.9304
VD2D[**?**]	0.9174	0.9771	0.9463
VD2D3D[**?**]	0.9555	0.9891	0.9720
Ours(1-stage)	0.9802	0.9576	0.9688
Ours(2-stage)	**0.9880**	0.9759	0.9819
Ours(3-stage)	0.9815	**0.9917**	**0.9866**

4.1.3 ISBI 2012 EM Segmentation Dataset

The other dataset for neuronal boundary detection in EM images is the ISBI 2012 EM Segmentation Dataset. As the dataset for the popular ISBI 2012 EM Segmentation Challenge, ISBI 2012 EM Segmentation Dataset does not offer the ground-truth segmentation for the testing stack, which is reasonable but increases the difficulty of results analyzing and discussion. Besides, the raw training data with annotations are relatively insufficient, where there is only one stack for training with the dimension of $30 \times 512 \times 512$. To train a very deep network using such a few images is theoretically difficult, even though we use the standard data augmentation to enrich the training materials.

Network settings are the same as ones we used in the experiment for Mouse Piriform Cortex Dataset[**?**]. Standard data augmentation (Section 4.1.2) is considered here, too.

The Rand F-score data from the leaderboard of ISBI 2012 EM Segmentation Challenge are shown in Table **??** . Not all the scores are presented since there are many entries in the leaderboard which have not been reported in literature. It worths noting that most of the leading methods

prcurve.png

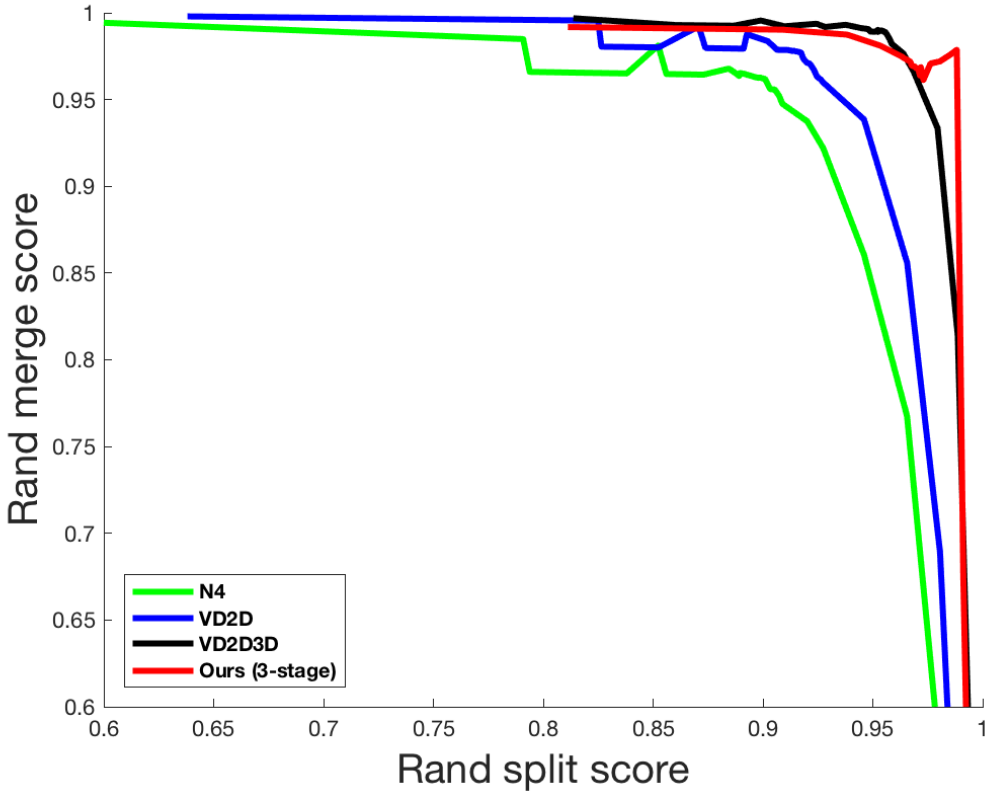

Figure 4.2 Precision (rand merge)-recall (rand split) curves on Mouse Piriform Cortex Dataset[**?**]. Our 3-stage network outperforms all the previous works on this dataset.

apply post-processing algorithms or assemble several models to improve the ranking. However, the proposed cascaded networks without any post-processing procedures can achieve competitive result to those with post-processing. The post-processing used in each approach is also presented in Table **??**, where [**?**] is a post-processing algorithm itself and can be adopted to ours. With a proper post-processing, our network can win the state-of-the-art with more powerful ResNet[**?**] as its base network. In the future, we will embed the ResNet[**?**] in our cascaded framework to further improve the performance.

Figure 4.3 Examples selected from PASCAL VOC Contour Dataset[**?**]. There are various objects (human, artificialities, animals, plants, etc.) appearing in complex scenes (indoor and outdoor environments, colorful backgrounds, blurs, textural confusions, etc), which significantly increases the difficulty of detecting the object-level boundary.

4.2 Object Boundary Detection in Natural Images

Our work begins at Neuronal Boundary Detection of EM images but dose not end in it. By applying the proposed multi-stage network to common object boundary detection in natural images, we evaluate the generalization capacity of our network.

4.2.1 Metrics

Precision and Recall are the well-accepted metrics for two-class annotation tasks[**?**, **?**]. Given the notations as follows: (1) True Positive *(TP)*: the times of annotating a positive pixel as positive; (2) False Positive *(FP)*: the times of annotating a negative pixel as positive; (3) True negative *(TN)*: the times of annotating a negative pixel as negative; (4) False positive *(FN)*: the times of annotating a positive pixel as negative; we have Precision *(P)*, Recall *(R)* and their harmonic average F-measure

$$P = \frac{TP}{TP+FP} \tag{4.5}$$

$$R = \frac{TP}{TP+FN} \tag{4.6}$$

$$F-measure = \frac{2PR}{P+R} \tag{4.7}$$

$$\tag{4.8}$$

The Precision-Recall Curve (PR-curves) can be also generated by varying the threshold for boundary binarization[**?**].

4.2.2 PASCAL VOC Contour Dataset

PASCAL VOC Segmentation Datasets are a series of well-accepted object segmentation datasets, including the well-annotated natural photos released from 2007 to 2012. [**?**] apply DenseCRF[**?**] to further refine the boundary annotation from the segmentation ground-truth and release a large instance-level object boundary detection dataset, PASCAL VOC Contour Dataset, with 10582 training images, 1449 testing images and their corresponding boundary annotations. Different from EM images, these natural photos are taken from various scenes and contain kinds of objects (Fig. 4.3). For example, in each photo, the image scale, the number and category of the object are different. There are also many samples with blurred object boundary and confused background on the dataset. All these significantly increase the difficulty of boundary detection on the PASCAL VOC Contour Dataset, while call out a deep network with sufficient capacity to extract the highly abstract object-level features from the data.

With such a large dataset, the standard data augmentation with a ratio of 36 is not necessary nor efficient. We follow the light-weight data augmentation mentioned in [**?**], by randomly cropping four 224 $\times$ 224 patches from the raw image and flipping them once, resulting in 8$\times$ training samples. Here, the cropping is mainly for memory efficiency[**?**] and the mini-batch based training. Original FCNs[**?**] is not available for the training with multiple training samples in one mini-batch,

because these samples may have different scales and can result in error. By fixing the cropped patches into one-size, we are able to enlarge the batch-size, 8 in practice, for a more effective and faster training.

We fine-tune the proposed network on PASCAL VOC Contour Dataset using the same strategy in Section 4.1.2. Firstly, we train a single-stage network with the same structure as HED[**?**]. Then we use the fine-tuned HED network to initialize the training for our 2-stage and 3-stage cascaded fully convolutional networks. F-measure comparisons are shown in 4.3.

For optimization, we use SGD[**?**]. Other hyper parameters such as the level of convolution in each subnet, learning rate, momentum and weight decay are same as what we used in the experiments of neuronal boundary detection. As demonstrated in [**?**], to process all the training samples, named one epoch, takes around 10000 iterations. We train the single-stage, 2-stage and 3-stage networks for 2 epochs, 4 epochs, and 5 epochs respectively. Compared with the CEDN proposed in [**?**], the training for our cascaded networks costs only half of the time for training CEDN, which proves the effectiveness of the proposed recursively end-to-end training.

As we expected, traditional methods for local edge detection such as SCG, its multi-scale version MCG[**?**], and Structured Random Forest for Edge Detection *(SE)*[**?**] present poor results on this benchmark. These methods use hand-crafted features which often combine multiple cues like color, brightness, spectrum, to handle different cases. HED[**?**] has the ability to naturally obtain the hierarchical features due to the adoption of the holistic-nested network, which is also used in our sub-networks. We fine-tune it on PASCAL VOC Contour Dataset and receive a F-measure of 0.52, much better than the traditional methods but lower than our 2-stage network. What's more, our 3-stage network outperforms CEDN[**?**] by 2% and achieves the state-of-the-art score on this benchmark. PR-curves in Fig. 4.4 illustrate the refinement benefits from the cascaded network structure.

Table 4.3 Object boundary detection evaluation comparison on PASCAL VOC Contour Dataset[**?**]. Our proposed 3-stage cascaded fully convolutional network achieves the new state-of-the-art on this benchmark, with a significant improvement (around 2% over the second)).

	F-measure
SCG[**?**]	0.36
MCG[**?**]	0.37
SE[**?**]	0.37
HED[**?**]	0.52
Ours(2-stage)	0.53
CEDN[**?**]	0.57
Ours(3-stage)	**0.59**

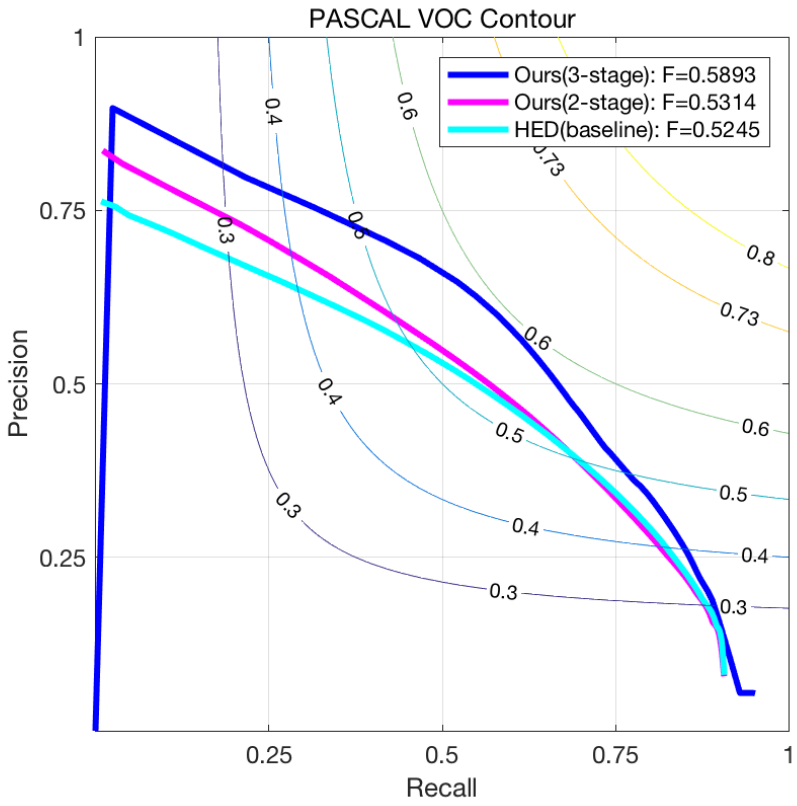

Figure 4.4 Precision-recall curves on PASCAL VOC Contour Dataset[**?**].

Chapter 5

Conclusions

In this paper, we introduce a novel text classification model called Stacked Residual Recurrent Neural Network with word weight. This model is able to extract more features and learn high hierarchical meaning of each word in a sentence. It also can identify the importance of different words due to its word weight structure. The residual structure makes model more expressive when stacked layers are deep. Experimental results show that our model can achieve high performance on text classification task than any other methods. This suggests our model can capture more potential features in sentences.

Bibliography

[1] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[2] H. Brücher, G. Knolmayer, and M.-A. Mittermayer, *Document classification methods for organizing explicit knowledge*. Institut für Wirtschaftsinformatik der Universität Bern, 2002.

[3] O.-W. Kwon and J.-H. Lee, "Text categorization based on k-nearest neighbor approach for web site classification," *Information Processing & Management*, vol. 39, no. 1, pp. 25–44, 2003.

[4] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents." in *ICML*, vol. 14, 2014, pp. 1188–1196.

[5] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[6] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[7] R. Johnson and T. Zhang, "Semi-supervised convolutional neural networks for text categorization via region embedding," in *Advances in neural information processing systems*, 2015, pp. 919–927.

[8] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[9] Z. Xiao and P. Liang, "Chinese sentiment analysis using bidirectional lstm with word embedding," in *International Conference on Cloud Computing and Security*. Springer, 2016, pp. 601–610.

[10] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1422–1432.

[11] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, 2015, pp. 2267–2273.

[12] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Learning distributed word representations for bidirectional lstm recurrent neural network," in *Proc. of ICASSP*, 2016.

[13] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, pp. 806–814, 2016.

[14] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[15] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[18] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[19] J. Chung, C. Gülçehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," *CoRR, abs/1502.02367*, 2015.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631. Citeseer, 2013, p. 1642.

[22] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 2002, pp. 1–7.

[23] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[24] O. Irsoy and C. Cardie, "Deep recursive neural networks for compositionality in language," in *Advances in Neural Information Processing Systems*, 2014, pp. 2096–2104.

[25] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[26] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[27] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.