DEEP LEARNING BASED UNSUPERVISED TISSUE SEGMENTATION IN HISTOPATHOLOGICAL IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Troya Çağıl Köylü
November 2017

DEEP LEARNING BASED UNSUPERVISED TISSUE SEGMENTATION IN HISTOPATHOLOGICAL IMAGES
By Troya Çağıl Köylü
November 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Çiğdem Gündüz Demir(Advisor)

Hamdi Dibeklioğlu

Pınar Karagöz

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

DEEP LEARNING BASED UNSUPERVISED TISSUE SEGMENTATION IN HISTOPATHOLOGICAL IMAGES

Troya Çağıl Köylü
M.S. in Computer Engineering
Advisor: Çiğdem Gündüz Demir
November 2017

In the current practice of medicine, histopathological examination of tissues is essential for cancer diagnosis. However, this task is both subject to observer variability and time consuming for pathologists. Thus, it is important to develop automated objective tools, the first step of which usually comprises image segmentation. According to this need, in this thesis, we propose a novel approach for the segmentation of histopathological tissue images. Our proposed method, called *deepSeg*, is a two-tier method. The first tier transfers the knowledge from AlexNet, which is a convolutional neural network (CNN) trained for the non-medical domain of ImageNet, to the medical domain of histopathological tissue image characterization. The second tier uses this characterization in a seed-controlled region growing algorithm, for the unsupervised segmentation of heterogeneous tissue images into their homogeneous regions. To test the effectiveness of the segmentation, we conduct experiments on microscopic colon tissue images. Quantitative results reveal that the proposed method improves the performance of the previous methods that work on the same dataset. This study both illustrates one of the first successful demonstrations of using deep learning for tissue image segmentation, and shows the power of using deep learning features instead of handcrafted ones in the domain of histopathological image analysis.

Keywords: Deep learning, convolutional neural networks, transfer learning, histopathological tissue image segmentation, seed-controlled region growing.

ÖZET

HİSTOPATOLOJİK GÖRÜNTÜLERDE DERİN ÖĞRENME TEMELLİ ÖĞRETİCİSİZ DOKU BÖLÜTLEMESİ

Troya Çağıl Köylü
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Çiğdem Gündüz Demir
Kasım 2017

Günümüz tıbbında, dokuların histopatolojik incelenmesi, kanserin teşhisinde önemli rol oynamaktadır. Fakat bu işlem, hem gözlemci değişkenliğine açık, hem de patologlar için zaman alıcıdır. Dolayısıyla, otomatik nesnel araçların geliştirilmesi önemlidir. Bu araçların ilk basamakları genellikle görüntünün bölütlenmesidir. Bu ihtiyaca yönelik olarak, bu tez çalışmasında, histopatolojik doku görüntülerinin bölütlenmesi için, *deepSeg* adını verdiğimiz iki aşamalı metodla, yeni bir yaklaşım sunuyoruz. İlk aşama, AlexNet adlı ve medikal olmayan ImageNet alanında eğitilmiş evrişimli yapay sinir ağında saklanan bilgiyi, medikal alanda bulunan histopatolojik doku görüntüsü karakterizasyonu için aktarmaktadır. İkinci aşama ise bu karakterizasyonu, çekirdek kontröllü bir bölge büyütme algoritmasında kullanarak, heterojen doku görüntülerini homojen bölgelerine öğreticisiz olarak bölütlemektedir. Bu bölütlemenin doğruluğunu test etmek için, mikroskobik kolon dokusu görüntülerinde testler yapılmıştır. Elde ettiğimiz sayısal sonuçlar, önerdiğimiz metodun, aynı veri kümesi üzerinde çalışan diğer metodların performanslarını arttırdığını göstermiştir. Bu çalışma, hem derin öğrenme tekniklerini doku görüntü bölütlemesi için kullanan ilk başarılı örneklerden biri olarak yerini almış; hem de histopatolojik görüntü analizinde derin öğrenme temelli özniteliklerin, elle çıkarılanlara üstün geldiğini göstermiştir.

Anahtar sözcükler: Derin öğrenme, evrişimli yapay sinir ağları, aktarmalı öğrenme, histopatolojik doku görüntü bölütlemesi, çekirdek kontrollü bölge büyütmesi.

Acknowledgement

I primarily want to thank my advisor Çiğdem Gündüz Demir. Not only she determined the direction of this thesis study, she also provided key guidance and continuous help that made it possible for me to finalize this study. I cannot envisage how this study would be possible without her. Likewise, I would also like to thank my thesis committee members Hamdi Dibeklioğlu and Pınar Karagöz. Their valuable comments helped the improvement of this study and lead it to its final state. I also cannot forget my colleagues in the Computer Engineering department. Starting from my lab piers Can Fahrettin Koyuncu and Simge Yücel, I want to thank all my friends in the department for their invaluable support and friendship.

By completing this thesis study, I come to an end of a phase in my educational life. It has been around seven years since I started studying in Bilkent University. My undergraduate study in Electrical and Electronics Engineering and my masters study in Computer Engineering helped me to gain numerous attributes; both academic and social. I therefore want to thank all the students, staff and faculty such as Ebru Ateş, Ayhan Altıntaş, Erman Ayday, Varol Akman, Daniel DeWispelare, Engin Soyupak, İlgi Gerçek; and all the friends that I met here. I feel very privileged to have been here for a such long time.

Of course, I would like to thank the people who helped me to obtain this privilege, namely my previous teachers such as İrfan and Şebnem Cantürk, my acquaintances from TED Ankara College high-school, my family and especially my mother Meltem Keskin. Without her support and determination for my education throughout my life, I would not be able to achieve any of this. So it is possible to say that she is the real architect behind anything I achieve.

With the eventuality of this thesis study, I feel the joy of making a dribblet of contribution to the scientific literature. This was all possible due to visionaries such as İhsan Doğramacı and Mustafa Kemal Atatürk, who valued knowledge and science over all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the diagnosis and treatment selection of cancer, accurate interpretation of pathological results plays an important role [1, 2]. With the development of technology, the variety in its treatment has increased; different diagnosis results in a different treatment selection. However there may exist disagreements on the diagnosis among pathologists. Depending on the case, the overall process may be significantly complex and subjective. Additionally, there is a significant increase in the number of cases. Digital pathology on the other hand provides faster and more objective decisions, and thus, they start to play an important role as a diagnostic tool [3].

The typical first step of such a tool is segmentation, which aims to separate regions of different properties in a histopathological image. In this study, for facilitating the diagnosis of colon adenocarcinoma, a cancer type which accounts for 90-95% of all colorectal cancers, we present a deep-learning based two-tier unsupervised segmentation method, which we call *deepSeg*. In its first tier, we use highly effective deep convolutional neural networks (CNN) to learn the characterization of tissue formations. In this tier, we use the available information previously learned from a vast collection of natural images by using a technique called transfer learning. In the second tier, we use the obtained characterizations to divide a histopathological image into its normal and cancerous regions.

This tier uses a seed-controlled region growing algorithm. Our overall end product method therefore is able to delineate the homogeneous regions in an unseen histopathological colon tissue image. To illustrate the effectiveness of our method, we conduct experiments on annotated images and compare the evaluation results with the previously developed methods. These results show that *deepSeg*, the method which we propose in this thesis, outperforms previous methods and it is a viable method to be considered in the future of computational pathology.

The overview of this chapter and others as follows; first, the motivations that drove us to devise our segmentation method and our resulting contributions to the literature are mentioned in this chapter. Next, in Chapter 2, background information necessary to follow the discussion in other chapters are presented, with the existing studies in the literature. In presenting the background information, a discussion about artificial neural networks is not included; this topic is assumed to be readily available. Rather, convolutional neural networks are presented as they emerged much more recently. In Chapter 3, our two-tier methodology is presented. Lastly in Chapter 4, the experimental performance of our method is shown in comparison with the other methods, with also including information about the experimental dataset. Analysis and discussion about the results, work done and potential future work concludes the thesis.

1.1 Motivation

In commencing this study, we have two main motivations as driving forces. These are:

- Insufficient usage and exploration of deep learning systems in the domain of digital pathology and especially for its segmentation problem, although they were proven to be effective in many computer vision tasks.
- The need for a top performance system in the domain of pathology that constantly becomes more and more complicated.

The first motivation is explored in more detail when the previous related work is introduced in Section 2.3. However, as a foreword, it can be stated that the use of deep learning and convolutional neural networks in the domain of digital pathology is rare, where especially convolutional neural networks are shown to work effectively in various image related tasks [4]. The second motivation follows the first one, where we believe that using these networks for the purpose of segmentation will both improve upon the results of previous studies, and thus, will form a basis for further improvements. This kind of an improvement is necessary to overcome additional issues created by the never ending development of technology and complication in the domain. Likewise, we also feel that any novel learning system that becomes the state-of-art in image processing must be employed in digital pathology for the hopes of gaining better results, in this case; convolutional neural networks.

1.2 Contribution

Following from our motivations, the main contributions of this study are twofold. First, our proposed *deepSeg* method uses deep learning features to characterize histopathological images. Majority of the studies dedicated to image segmentation in this domain use hand-crafted features, which makes them highly task-specific. Furthermore, in obtaining deep learning based features, we transfer knowledge from a non-medical domain, which is again one of the first such applications. With this study, we showcase the effectiveness of using convolutional neural networks in image related tasks, and their ability to learn image characterizations that enables more accurate image processing.

Our second contribution is related to the way we process histopathological images once we obtained deep-learning based characterizations. Our use of this characterization in segmentation is different than the standard approach employed by the previous studies. After dividing the image into its pixels or equal-sized patches that contain a collection of pixels, these studies usually define class labels for their pixels/patches and use these labels for segmentation. In other

words, in these studies, there is one-to-one correspondence between class labels and region types. They assign each pixel/patch to one of the classes and form connected components with the ones that belong to the same class label. The segmentation that they obtain is a result of this straightforward operation. However, this approach can be problematic, because there can be some small regions that are visually different than their regions of interest, meaning that they do not and should not affect the overall region type. Such examples can be seen in Figure 1.1, namely the regions labelled with **A** and **B** (see the figure for detailed explanation). Simply defining a class label per region type and using class labels for segmentation in these cases raises additional issues as training a classifier becomes harder, and the trained classifier may mislabel these pixels/patches that correspond to those subregions. Thus in our proposed method, our classifier learns more classes (including normal and various cancer types) than the types of the regions to be segmented. This extended characterization is used in a region growing algorithm, which is able to use more information to determine patch-region similarity that the region will grow upon, rather than just connecting patches/pixels with the same label.

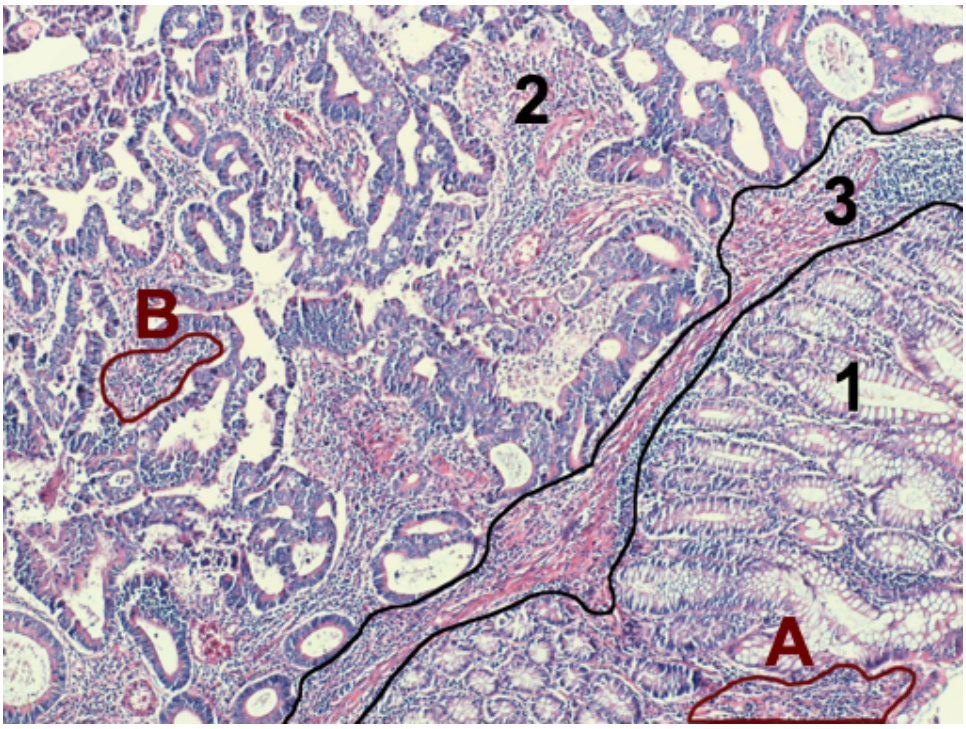


Figure 1.1: Sample tissue image for illustrating adenocarcinoma. This work focuses on unsupervised segmentation of colon images that contain normal and adenocarcinomatous regions. Colon adenocarcinoma accounts for 90-95% of all colorectal cancers, thus being the most common form of cancer in colon tissues. This cancer type originates from glandular epithelial cells and causes deformations in glands. Nonglandular stromal regions that are found in between glands, are irrelevant in the context of colon adenocarcinoma diagnosis. On the figure, such a subregion (marked with 3) can be included in either a normal (marked with 1) or a cancerous (marked with 2) region, without changing the region's type. Furthermore, such nonglandular subregions (marked with A and B) are found in both normal and cancerous regions.

Chapter 2

Background

The study in this thesis is constructed on a two-tier operation; unsupervised segmentation of histopathological tissue images based on supervised learning of a deep convolutional neural network. The aim of this chapter is to include any necessary background information that will be referred in discussing our methodology. The structure is as follows; first, deep convolutional neural networks are reviewed (Section 2.1). Then, for the use of the special training method, transfer learning in the context of deep convolutional neural networks is examined (Section 2.2). Lastly, related work on the subject is mentioned (Section 2.3).

2.1 Deep Convolutional Neural Networks

Starting with the McCulloch-Pitts artificial neuron model, there have been numerous attempts to capture biological computation, which is the essence of human intelligence [5]. Now, neural networks play an important role in the field of machine learning by solving numerous tasks. An emerging issue however; there is *no-free-lunch* in learning (artificial) intelligence tasks. This means that, there is no single algorithm or model that captures the nature or essence of all possible situations [6, 7]. For standard artificial neural networks (these networks are also

called as *fully connected neural networks*, due to the fact that all inputs of their layers are connected to all neural/computational elements) this problem emerges for data of large sizes, like images. To capture the nature of large data, a straightforward application is to increase the depth and width of the network. But when this is done, obtained performance is most generally very poor, where training data evaluation performance is high but test data (the data that was not used during training) evaluation performance is poor. This is a phenomenon called *overfitting*, and it is related to memorization rather than learning.

To alleviate this problem for image related tasks, a specialized artificial neural network model was devised. Taking its inspirations from biological sensory neurons [8], the deep convolutional neural network (DCNN or CNN for short) was proposed. This model can be thought as a (generally) deeper neural network. Like regular neural networks, CNNs must be trained with training data using (typically) backpropagation learning algorithm. After the training phase, the forward-pass operation of this network can be used to evaluate unseen data. The problems arising with increasing depth are solved by using specialized layers for hidden layers, before the regular fully connected ones. One of the most important properties of some of these layers is that they greatly reduce the number of parameters that must be learned. This reduction is an important solution to many learning based issues if done efficiently. Just like a regular neural network, a CNN is a trainable black-box estimation and the important types of layers employed are explained here. Note that each of these layers can be thought as volume transformers, where they transform an input volume into an output. Some layers alter the input volume size, while others do not. An example is shown in Figure 2.1.

The sample CNN shown in Figure 2.1 is composed of three convolutional layers and two pooling layers, whose details will be explained shortly. One aspect visible from the image is the transformation of the volume. For instance, the input volume of size 28×28 is transformed into a volume of size $24 \times 24 \times 4$ by the convolutional layer. And the pooling layer transforms this volume into a volume of size $12 \times 12 \times 4$. At the end, the aforementioned data size reduction can be observed– namely, a proceeding fully connected layer must process 26 elements rather than 28×28. In this understanding, now, the detailed information on

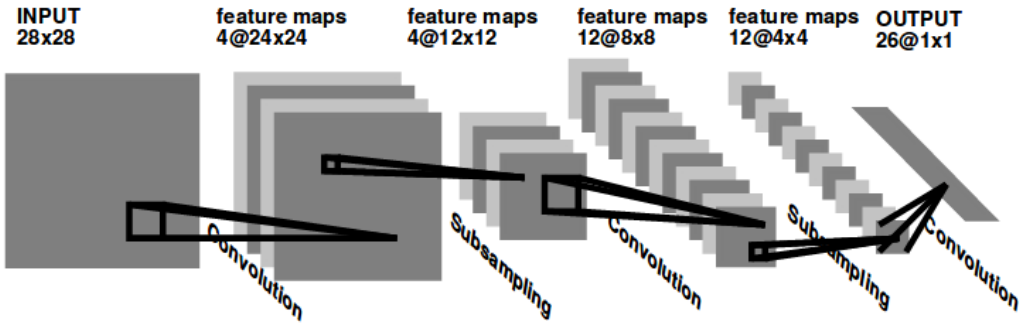


Figure 2.1: A sample CNN model from [9]. This image is used under the permission of the copyright owner: The MIT Press.

individual layers are presented.

- **Convolutional layer:** This layer is the most essential layer of a CNN. The inspiration for this layer is from the brain anatomy, as it is thought that some layers of sensory neurons perform filtering on their input. Therefore this layer performs filtering, in other words, convolution on its input and learned filter. Apart from the biological connotation, this layer is very efficient in keeping the number of variables to be learned in acceptable levels. A non-specialized fully connected layer has to independently learn all its *weight* variables, the number of which is $H \times W \times Z \times L$; where H, W, and Z are input dimensions and L is the number of neurons in the layer. This is necessary as all neurons are connected to all inputs, and there is a weight parameter for each of them. A convolutional layer on the other hand, uses shared weights as filters. The filter is the same for all input patches, and it can be thought as if it slides through the input area. As the training operation proceeds, calculated updates for the same filter positions are aggregated and applied as the same to result in again same weights. Therefore, the number of independent variables to be learned are reduced to $H_k \times W_k \times N$; where H_k, and W_k are filter dimensions and N is the number of filters. As expected, the latter is much smaller– in our study for instance, images that are inputted to the first (convolutional) layer of our network are of size $227 \times 227 \times 3$ (so naturally L should also be large in

a fully connected layer), but the filters that we use are of dimensions than $11 \times 11 \times 3$ and there are 96 of them.

The output of this layer can be regarded as a volume of filtering outputs for different filters, merged on the third dimension. The obtained volume after this layer's operation will be of size $[(H - H_k + 2P)/S + 1] \times [(W - W_k + 2P)/S + 1]$, where P is *pad*– the parameter used to determine the amount of pixels that are added to the borders of the input (with techniques such as mirroring and zero-padding) and S is *stride*– the parameter that determines the number of pixels that the sliding window slides after each iteration of the convolution. A common practice is to select filter dimensions equal (H_k and W_k). Another common practice is to select P and S such that the obtained output is the same size with the input, in height and width. One last point in this layer is; a filter's depth matches the depth of the input. For instance, if the input of the layer is an RGB image, the filters of this layer are of depth three. At each sliding iteration, a three dimensional convolution is applied, where each corresponding input and filter entry are multiplied and their results are added to make a single number.

- **Nonlinearity layer:** This layer is inspired from the nonlinear input-output behavior of the biological neuron. This layer maps the input to a limited real range nonlinearly. A direct benefit is thus the limitation of the numeric range. Most common used types of nonlinear functions in this layer are *Sigmoid* and *Rectified Linear Unit* ($ReLU$). Sigmoid function is defined as $S(x) = \frac{1}{1+e^{-x}}$. ReLU is defined as $f(x) = max(0, x)$. As the only operation accomplished by this layer is the mapping of input to an output depending on the funciton in use, there is no learning needed.

- **Pooling layer:** This layer can be viewed as a compressor of information. Again for this layer, stride and kernel size parameters must be selected, just like the convolutional layer. With these selections, a sliding window which has the size determined by the kernel is traversed through the input volume. At each iteration, one number is produced over an area. Then the stride parameter is used to determine the slide of the window. According to these parameters, the input volume is converted to a much smaller output volume.

The common usage is to select these parameters to create a non-overlapping slide, such that the output surface area is four or sixteen times smaller than the input area. An important point here is that the sliding windows in this layer are two-dimensional, where convolutional layer windows are typically three-dimensional (whose third dimension can be of any size). Therefore, the output volume of this layer has the same depth with its input's volume.

There are two common modes of operation for this layer; *maximum* and *average* pooling. Maximum pooling compresses the information by taking the maximum value at each sliding window iteration, whereas average pooling takes the average of the values inside the window. Again note that there is no learning required in this layer.

- **Fully connected layer:** This layer is the basic neural network layer. For every neuron in this layer, a weight is assigned and learned for each input element. The operation of this layer can be viewed as an inner product between the input vector and the learned weight matrix.

In a typical CNN, one or more convolutional, nonlinearity and pooling layers are paired in this order, before one or more fully connected-nonlinearity layer pair. In a classification system, the last layer is a fully connected (decision) layer, with the number of outputs (or neurons) selected as the number of input classes. Here, each output signifies the class probability of the input (when the softmax function is used on the results of the decision layer). The class with the maximum probability is therefore selected as the class of the input. Another key difference between CNNs and regular neural networks is that in CNNs, the fully connected layer is inputted with a much smaller input size and thus, a much smaller number of fully connected layer parameters like neurons and weights are needed. Overall, this results in a limited number of independent variables that must be learned or adjusted in the training phase– a feature that helps to overcome overfitting.

2.2 Transfer Learning

One can employ different strategies when training a CNN on a dataset. Here, a number of them are presented. Note that these methods are not exhaustive; using other techniques, or using a mixture of these techniques are also viable.

1) **Learning from scratch:** When the dataset is adequately large, a training operation that starts on randomized weights can eventually avoid overfitting [10].

2) **No additional training:** A pre-trained CNN model can be used without any further training. Outputs of one of its layers can be treated as features, and they can be used to train other classifiers or evaluators, such as support vector machines (SVMs).

3) **Fine-tuning:** A pre-trained network can also be used in other ways than a feature extractor. One can take a pre-trained network, modify it or not, and start (or continue) training iterations on another dataset.

Transfer learning in deep learning is the transfer of knowledge between source and target models [11, 12]. In that context, note that the options two and three described here can be used to relay information between models. In this work, the fine-tuning option is employed (thus we use the words training and fine-tuning interchangeably to refer the learning process of our CNN). When a pre-trained model is selected for fine-tuning, there is no need to use the same dataset, or the exact model. A generally employed scheme is to take a model that was trained on an extensive image dataset, and resume training on a new but limited image dataset. Due to the weight sharing property of convolutional layers, there is no need to scale the new dataset images to the same size, or preserve all layers. Only data size preservation problem arises in the last fully connected layers. Therefore, a plausible strategy is to change and train incompatible last layer(s) from scratch, while training exported layers with a smaller weight update rate. This strategy is plausible because of the inner workings of a CNN– a pre-trained

CNN on natural images shows a phenomenon where early layers learn general and low level features such as Gabor filters and later layers learn task specific high level features [13]. Such kind of transfer learning has been successfully applied for many computer vision tasks; including image recognition [11], segmentation [14], and video classification [15].

2.3 Related Work

Existing studies on tissue segmentation generally rely on using handcrafted features. These studies typically partition an image into primitives (patches, pixels or objects) and they characterize these primitives using the handcrafted features. These primitives are then merged based on their extracted features to form segmentations.

Patch-based studies divide the histopathological images into equal sized patches. From these patches, they extract handcrafted color and texture features from the pixels inside these patches, and classify them based on these features. Then, they form the segmentation by connecting the patches of the same class. In this understanding, Mete *et al.* extract features from the results of color based clustering [16]. They train a support vector machine (SVM) on these features, and with the patches that this classifier identifies as positive, the delineation is obtained. Wang *et al.* extract statistical texture features from their slides [17]. Again by using a SVM, they obtain the segmentation as a result of a classification process consisting of multiple steps. Romo *et al.* split an image into a grid of blocks, where each block is represented by the feature spaces of color, intensity, orientation, and texture [18]. Target and distractor examples are picked for learning, and similarity metrics allow to find clusters of these examples in the feature spaces. A probability map is obtained by integrating similarity maps of each subspace, which allows for the selection of regions of interest.

Pixel-based studies' follow a similar approach. However, they omit the step of patch divide, and directly classify pixels using their handcrafted features. Then,

they merge the pixels of the same class to form segmented regions. Mosaliganti *et al.* use a three-step process where in the first step, they label each pixel belonging to one of the four microstructural components based on color [19]. Following this, they obtain correlations, which are eventually used as features for the final classification, that forms the segmentation. Signolle *et al.* uses wavelet-transform to subsample very large histopathological images. On these samples, they construct hidden Markov tree models for the classification of pixels [20]. This study also includes the merging of classifiers, which are obtained for different hidden Markov trees for different hyper-parameter selections.

Lastly, object-based studies represent an image with a set of objects that corresponds to approximate locations of histological tissue components. They characterize each object with the spatial distribution of other objects within the specified neighborhood. In these studies, the handcrafted features are obtained for object-level textures, which are used by a region growing algorithm to merge the objects for forming the segmentation. Tosun *et al.* obtain their objects by locating circular primitives on pixel collections that are grouped by color intensities [21]. The segmentation that follows is obtained based on defined measures of these objects, by using a region growing algorithm. In another study, again Tosun *et al.* introduce a texture measure to quantify the spatial relations of tissue components [22]. For that, this study defines a run-length matrix on constructed graph nodes, similar to defining a run length matrix on gray pixel values, and extracts texture features from this graph run-length matrix. Şimşek *et al.* define their objects similarly and quantify their distribution by defining the cooccurence matrix on this object set [23]. Then they use a multilevel graph partitioning algorithm that uses this quantification to achieve segmentation.

Only a few studies use deep learning based features to segment histopathological tissue images. In the study by Xu *et al.*, the overall aim is to segment an image into two regions of interest; epithelial and stromal [24]. It accomplishes this task using three handcrafted feature based methods versus variations of CNN feature based methods. The CNN method uses patch based processing (patches can be obtained by simply sliding a window or using other algorithms), paired with classifiers of softmax and support vector machines. In this study, patches are

directly assigned to their classification labels to obtain the final segmentation. In the end, this study shows that CNN based approaches outperform handcrafted feature based approaches. Also, Arevalo *et al.* use CNNs to detect basal cell carcinoma regions [25]. For this task, the presented method is a four step algorithm. In the first step, it uses autoencoders or independent component analysis to obtain a set of feature detectors for the input image. In the second step, the image is characterized with using the obtained feature detectors by using the CNN approach. Here, an interesting point is that this method does not use standard training for the CNN model it uses. The CNN model here is merely used for its forward pass operation, where there is only one convolutional layer and one pooling layer, and its convolutional layer uses the feature detectors obtained in the previous step as filters. Based on the extracted features, in the third step, it trains a binary classifier, that indicates whether cancer is present in an image or not. Finally in the last step, it uses visualization techniques to illustrate which regions in the image are related with cancerous patterns, which can also substitute for a segmentation.

Rather than these studies, there also exist other studies that focus on segmentation using deep learning. However, these ones do not focus on tissue segmentation, which aims to segment a heterogeneous tissue image into its biologically meaningful homogeneous regions. Instead, they focus on (segmenting) nuclei in medical images, which aims to differentiate nuclear objects from the background. First, Xing *et al.* use a CNN based method to segment nucleus regions from the rest of the medical image [26]. In this study, a CNN that is trained on patch size images is used to generate a probability map on pixels, which corresponds to how likely a pixel belongs to a part of a nucleus. Then, an iterative region merging algorithm is used on this map to mark the nuclei. Sirinukunwattana *et al.* have the aim of detecting nuclei, using a CNN [27]. The CNN is used to assign probabilities to each pixel being the center of a nuclei, in a sliding window manner to process patches. They then finalize the detection using local maxima in the probability map.

Finally, there is only one work to our knowledge that combines deep learning, transfer learning and medical image related tasks. Shin *et al.* use a couple of known CNN models that have been previously used, for detection and classification tasks in the medical domain [28]. Both transfer learning (including fine-tuning and feature usage without additional training) and training from scratch are used as training strategies on these models. The findings reveal that transfer learning from natural image datasets can be beneficial in medical image processing tasks.

Chapter 3

Methodology

Our proposed method relies on using deep learning for unsupervised tissue segmentation. This method, which we call *deepSeg*, is based on a two-tier operation. In Tier 1, a convolutional neural network (CNN) is used to learn the characterization of histopathological images. We transfer a CNN that is pre-trained on a natural image dataset with the technique defined in Section 2.2. When trained with homogeneous tissue images in a supervised manner (the details of the used images, namely, differences between homogeneous and heterogeneous images, will be explained shortly), this model characterizes its input image with six posteriors, corresponding to six classes. Tier 2 then uses learned characterizations to segment a heterogeneous image into its homogeneous regions in an unsupervised manner. To accomplish that, these heterogeneous images are divided into overlapping patches. Using the CNN model from the first tier, each patch is characterized by six posteriors. This characterization is initially used to determine homogeneous seed regions. These seeds are then grown using our region growing algorithm to encompass the whole image. Lastly, our post-processing algorithm finalizes the segmentation.

Before delving into the details of both tiers, it is beneficial to briefly mention the properties of the images that are used in the first and second tiers. In Tier

1, the images used to train the CNN model belong to a dataset called *AuxiliarySet*. This set contains patch size homogeneous images. The images are called as homogeneous, because one image contains normal or cancerous tissue of a single cancer type. In Tier 2, the images used for segmentation are from a dataset called *SegmentationSet*. These images are heterogeneous and they are much larger than the patch size. As heterogeneity implies, the same image may contain both normal and cancerous regions. The details of these datasets will be provided in Section 4.1.

3.1 Tier 1: Transfer Learning from AlexNet to Supervised Homogeneous Tissue Image Classification

The aim of this tier is to obtain a CNN model that fairly characterizes homogeneous histopathological images. For this purpose, this tier transfers the knowledge from a pre-trained CNN model to our domain in a supervised manner, by fine-tuning (or further training) this CNN. Figure 3.1 shows the desired operation obtained at the end of this tier. In the following subsection, we first elaborate on the training procedure of the CNN, and then provide a collection of visual results obtained at the end of this tier's operation.

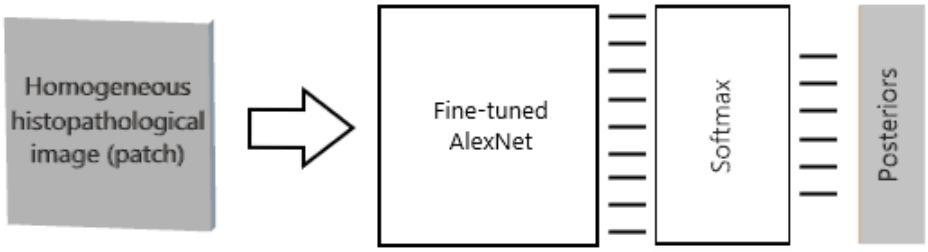


Figure 3.1: The end product of Tier 1. The desired CNN model after fine-tuning is expected to produce six posteriors that fairly characterize a homogeneous histopathological image. The decision layer indicated by "Softmax" is trained from scratch, to work as a classifier on top of the transferred information from previous layers of fine-tuned AlexNet. Details are specified in Section 3.1.1.

3.1.1 CNN Model Training

The CNN model that we use to train or fine-tune is AlexNet [29], that was originally trained for the ImageNet natural image dataset [4] for the challenge in 2010. The ImageNet dataset was constructed for an annual challenge where teams compete to train models to obtain highest evaluation accuracies in classifying unseen images. According to the numbers of 2010, there were approximately 15 million images and 1000 (high-level) classes. The AlexNet model achieved the best results in many classification categories over 1.2 million evaluation images. Their CNN consists of five convolutional layers, three max-pooling layers, and three fully connected layers to result in 60 million parameters and 650 000 neurons [29].

We use the Caffe [30] framework for CNN training. The AlexNet model they provide is a slight modification, that was trained on the ILSVRC12 challenge dataset for 310 000 iterations. The layers of the used model are given in Table 3.1, in their usage order. For completeness, the input, although not being a layer, is also included.

In Table 3.1, the first column specifies the type of layers. The second column supplies the values of the variables that characterize these layers. As can be seen, there are four different kinds of layers (excluding input information, which is self-explanatory). A convolutional layer is characterized by its filter kernel size, stride and pad it uses, and the number of outputs that it produces. A nonlinearity layer is only characterized by the nonlinear function it uses to produce its outputs. A pooling layer is characterized by its mode of operation, kernel filter size, and stride it uses. Lastly, a fully connected layer is characterized by the number of outputs it produces (more detailed information about CNN layers can be found in Section 2.1). Note that the decision layer in AlexNet produces 1000 posteriors, for the 1000 high-level ImageNet classes. Therefore, as the information learned in this layer is highly task specific; it must be omitted. This layer is learned from scratch– its weights are randomly initialized, and the number of outputs are set to six, which is the number of classes in our *AuxiliarySet*.

In this supervised fine-tuning with backpropagation, there are five parameters

Table 3.1: Essential layers of the used AlexNet model [29]. Other (utility) layers of varying kinds are omitted in this representation.

Layer Name	**Layer Information**
Input	$227 \times 227 \times 3$ (RGB) image
Convolutional	number of outputs: 96, kernel size: 11, stride: 4, pad: 0
Nonlinearity	type: ReLU
Pooling	type: MAX, kernel size: 3, stride: 2
Convolutional	number of outputs: 256, kernel size: 5, stride: 1, pad: 2
Nonlinearity	type: ReLU
Pooling	type: MAX, kernel size: 3, stride: 2
Convolutional	number of outputs: 384, kernel size: 3, stride: 1, pad: 1
Nonlinearity	type: ReLU
Convolutional	number of outputs: 384, kernel size: 3, stride: 1, pad: 1
Nonlinearity	type: ReLU
Convolutional	number of outputs: 256, kernel size: 3, stride: 1, pad: 1
Nonlinearity	type: ReLU
Pooling	type: MAX, kernel size: 3, stride: 2
Fully Connected	number of outputs: 4096
Nonlinearity	type: ReLU
Fully Connected	number of outputs: 4096
Nonlinearity	type: ReLU
Fully Connected	number of outputs: 1000

to be set. The first one is the base learning rate $base_\eta$, which is the initial rate of weight update. The second one is the multiplier γ, which is multiplied with the current learning rate η to obtain the new learning rate when a certain number of iterations passes. That certain number of iterations is the stepsize S. The next one is num_{iter}, the number of iterations required for the completion of training. The last one is the momentum α, which affects the weight update rate. For each weight, if the previous update was large, the current update becomes also large on account of this parameter. Else, it becomes smaller. In our experiments, we use the following values for these parameters: $base_\eta = 0.001$, $\gamma = 0.8$, $S = 2000$, $num_{iter} = 15000$, $\alpha = 0.9$.

The AlexNet model whose parameters are defined in Table 3.1 is illustrated in Figure 3.2. In this figure, the volume parameters are written next to appropriate positions. The first volume is the input, and max pooling operations are defined on the figure. The last three operations, where arrows are coded with the word

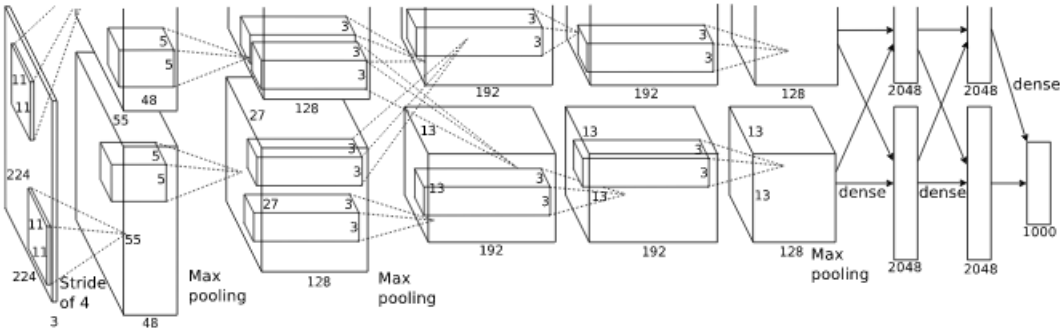


Figure 3.2: The AlexNet model defined in [29]. Used model parameters are given in Table 3.1. This image is used under the knowledge of NIPS Proceedings and permission of one of the copyright owner authors: Alex Krizhevsky.

"dense" belong to the fully connected layers. Other unmarked operations are convolutions. In this figure, the volumes are divided into two equal portions after the first convolutional layer. This division is intended to divide the operations related to top and bottom volumes into two graphics processing units.

As explained, in Tier 1 of our method, we start with the pre-trained AlexNet model and fine-tune its weights using the images of *AuxiliarySet*. Figures 3.3, 3.4, 3.5, 3.6, and 3.7 illustrate the learned weights (filters) of each of the five convolutional layers, respectively. Note that the first convolutional layer filters are three dimensional, so they are illustrated as RGB images. An important observation here is a lot among the first layer filters developed into orientations of Gabor filters. These filters are widely applied in image recognition tasks, as they extract orientation dependent frequency contents, such as edges [31]. This development coincides with the understanding that earlier CNN layers extract low-level features like edges. Starting with the filters of the second convolutional layer, the filters lose their connotation as they become more and more high-level and task specific. One last note is that relative dimensions of the filters among different figures are not to scale.

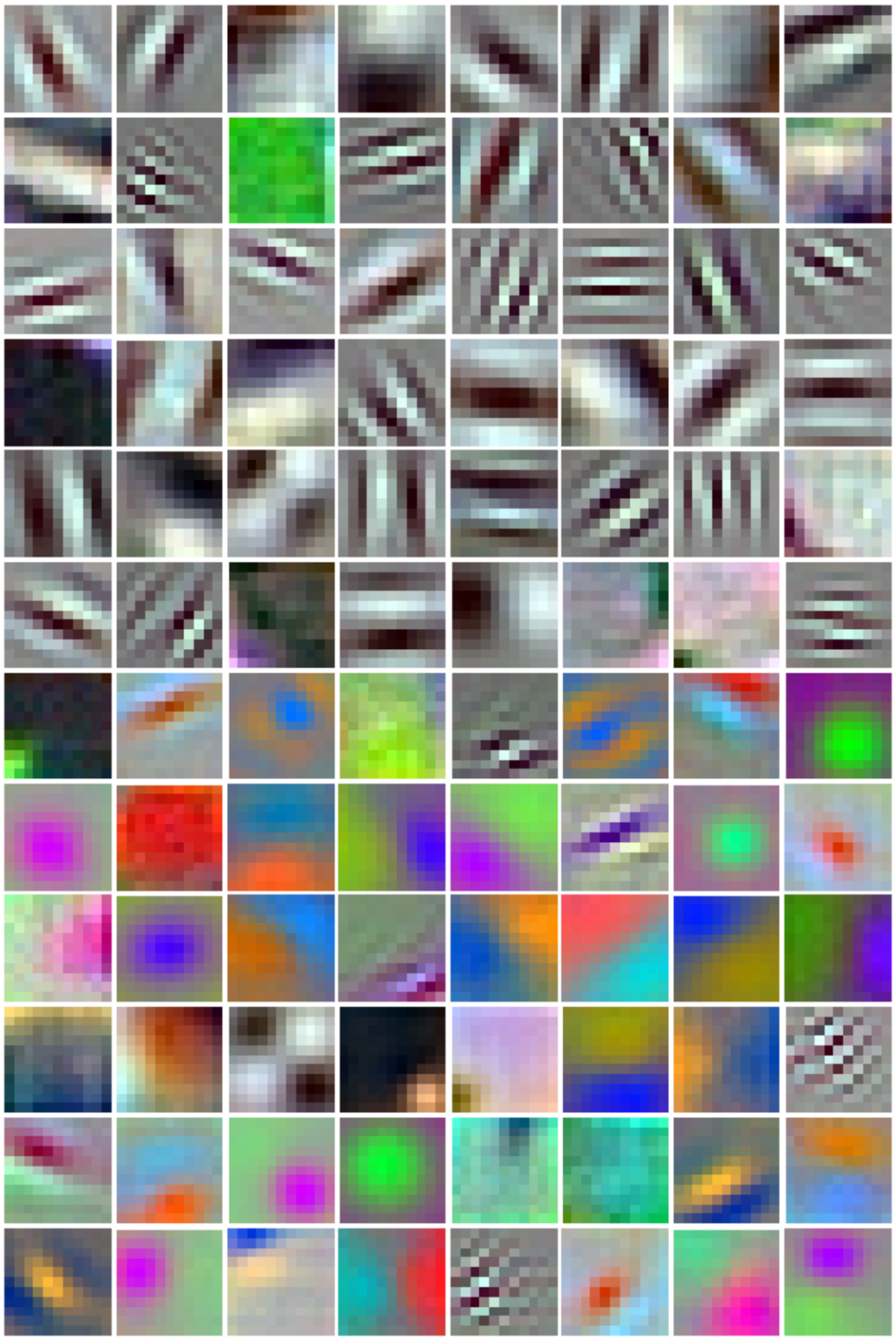


Figure 3.3: Resulting filters of the *first* convolutional layer, after fine-tuning. As the filter depth in this layer is three, the filters are illustrated as RGB images.

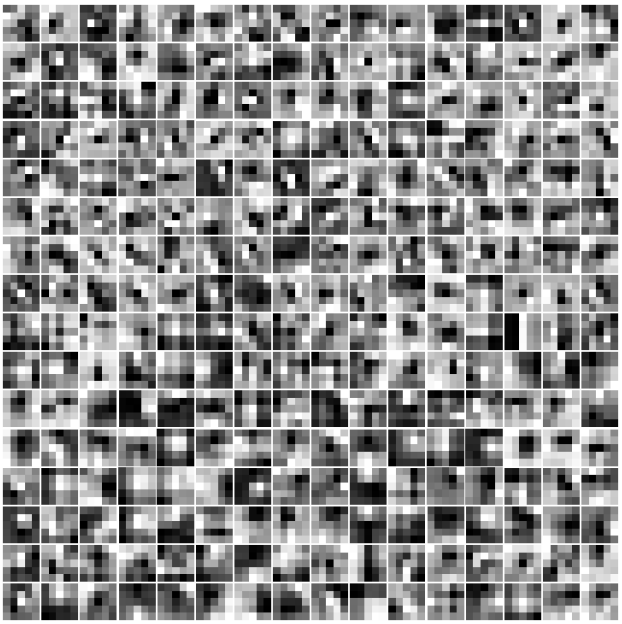


Figure 3.4: Resulting filters of the *second* convolutional layer, after fine-tuning. As the filter depth in this layer is larger than three, only the first dimension is illustrated.

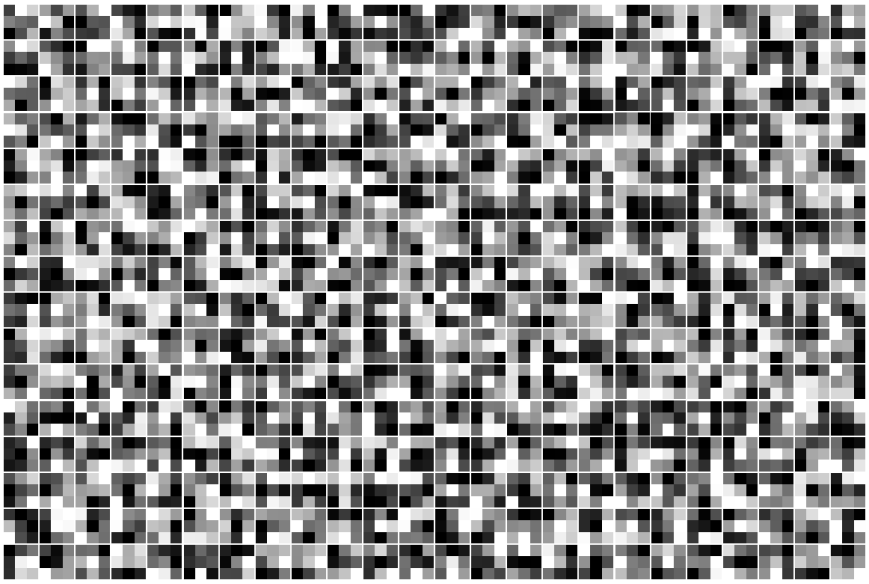


Figure 3.5: Resulting filters of the *third* convolutional layer, after fine-tuning. As the filter depth in this layer is larger than three, only the first dimension is illustrated.

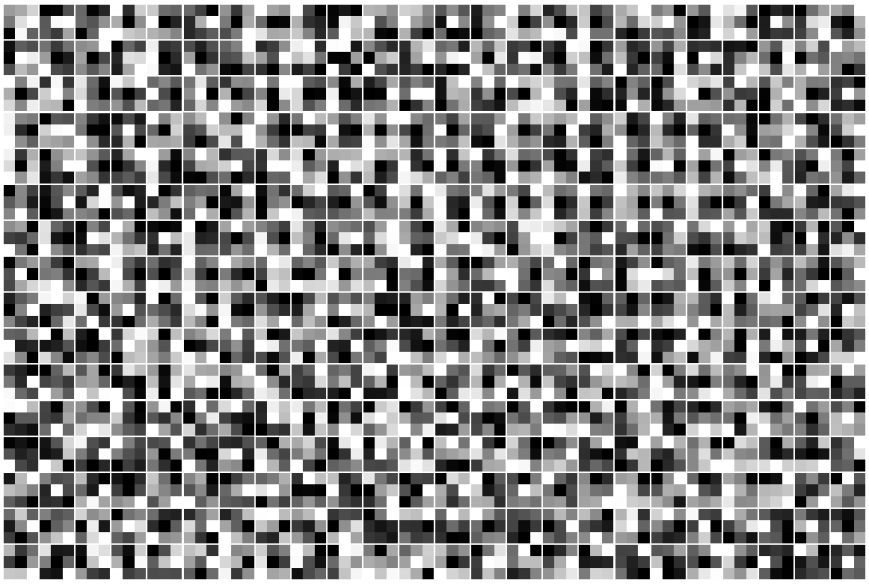


Figure 3.6: Resulting filters of the *fourth* convolutional layer, after fine-tuning. As the filter depth in this layer is larger than three, only the first dimension is illustrated.

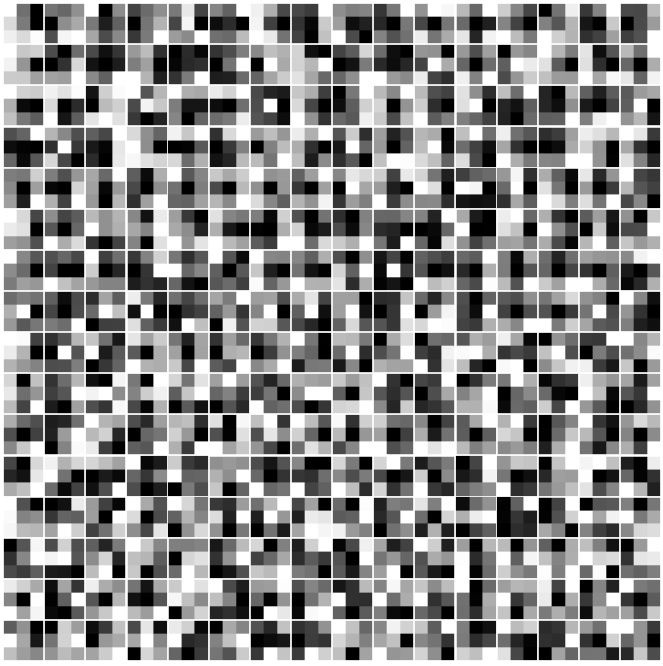


Figure 3.7: Resulting filters of the *fifth* convolutional layer, after fine-tuning. As the filter depth in this layer is larger than three, only the first dimension is illustrated.

3.2 Tier 2: Unsupervised Heterogeneous Tissue Image Segmentation from Supervised Homogeneous Tissue Image Classification

The aim of this tier is to segment a heterogeneous histopathological images into its homogeneous regions, in an unsupervised manner. The overall operation is summarized in Figure 3.8. This tier accomplishes this by a four-step seed-controlled region growing algorithm. First, the heterogeneous image is divided/decomposed into overlapping patches. The assumption here is that these patches are small enough to cover only a homogeneous content, and large enough to encompass a meaningful one. Using the trained CNN model from Tier 1, each of these patches is characterized by six posteriors (green box in Figure 3.8 and Algorithm 1). Using these posteriors as a map, we first identify homogeneous seed regions. They constitute regions that we are confident in using them as a basis for segmentation (yellow box in Figure 3.8 and Algorithm 2). Then, these seed regions are grown by an iterative region growing algorithm to cover the entire map, using the embedded posterior information (blue box in Figure 3.8 and Algorithm 3). Lastly, used mapping is converted back to encompass all pixels of the image by a patch-based voting. At the end of this remapping, some extra small regions may emerge. This tier thus ends its operation by merging these extra small regions, in other words, a clean-up is carried out (red box in Figure 3.8). The resulting segmentation is called *unsupervised*, because this tier only differentiates between regions in the end, there is no associated class label assignment. The following subsections contain the detailed information about the steps of this unsupervised segmentation.

3.2.1 Initialization and Seed Identification

The second tier starts by dividing the heterogeneous image (an example is given in Figure 3.9) into overlapping patches p_i, that is of size compatible with the trained CNN model in Tier 1. These patches are supplied to the CNN, and for each one,

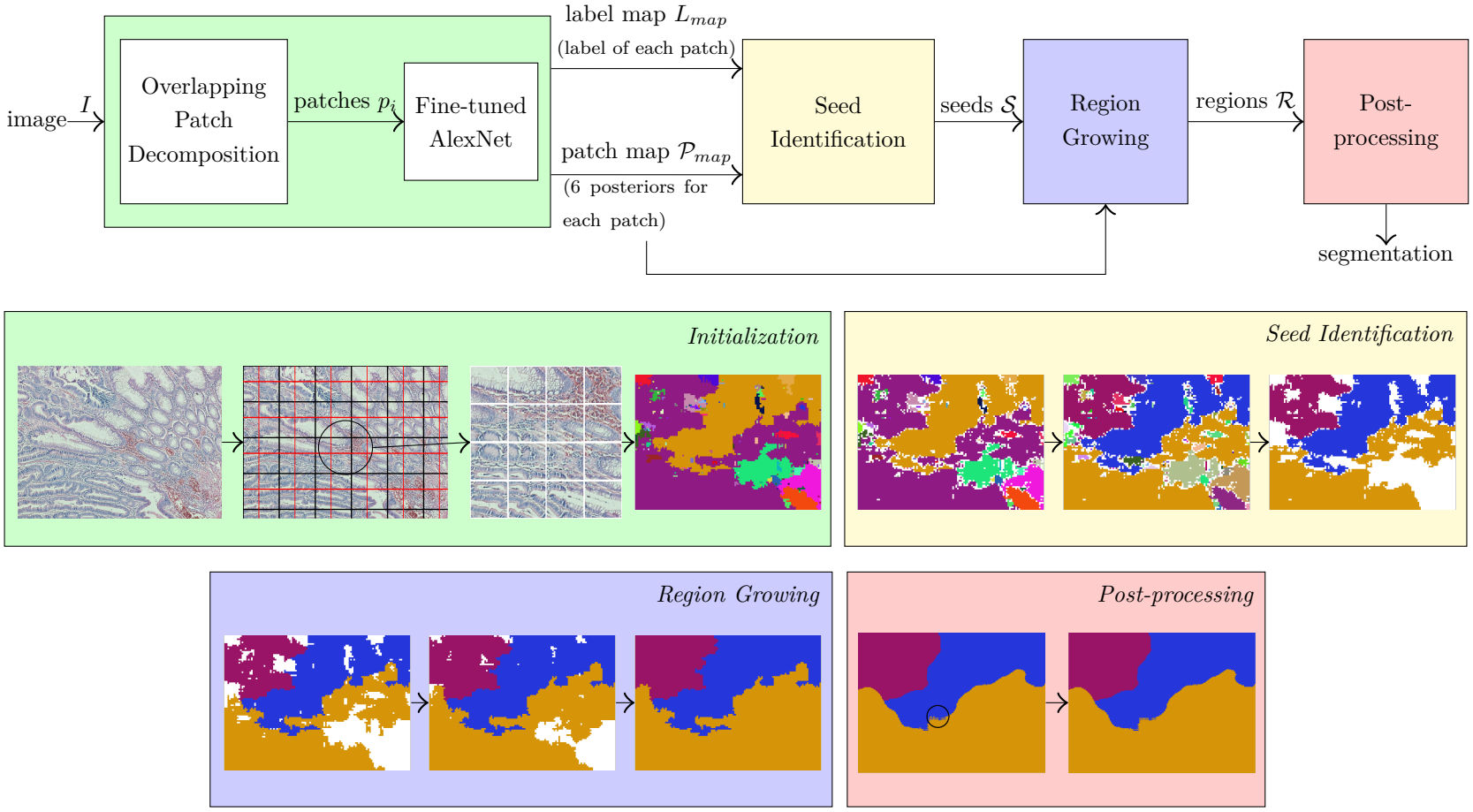
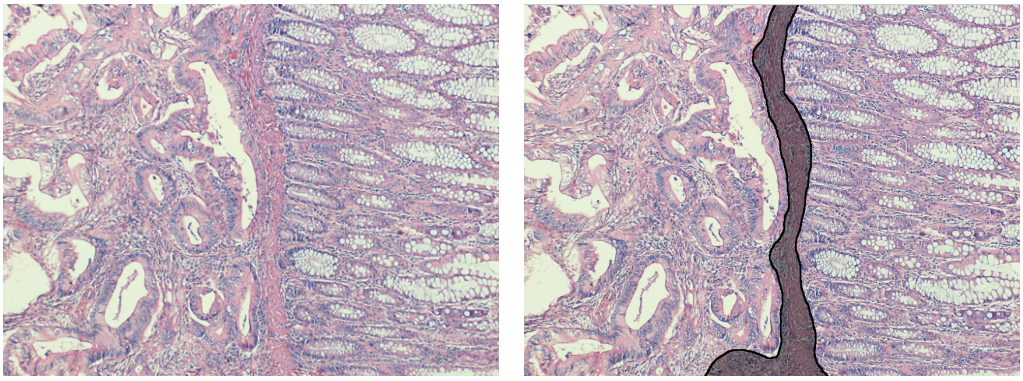


Figure 3.8: Schematic overview of Tier 2. Steps and terms are explained in the subsections of Section 3.2.



(a) (b)

Figure 3.9: A sample image from *SegmentationSet*: (a) Original image, (b) gold standard version that indicates homogeneous regions.

six posteriors $\{\pi_k(p_i)\}^6_{k=1}$ are calculated. According to these posteriors, each patch is labeled as $l(p_i)$ with the class corresponding to the maximum posterior value.

This initialization phase creates two maps over the image: the first one is the posterior map, $\mathcal{P}_{map}$, that is of size $m \times n \times 6$ and the second one is the label map, L_{map}, that is of size $m \times n$. Here, m and n are values that depend on parameters, including patch size (defined by $height_P$ and $width_P$), image resolution (defined by $height_I$ and $width_I$) and overlap distance d. In cases where the combination of these parameters do not result in a divisible patch number, the last patch takes the respective end points of the image, and encompasses a patch size by taking the necessary number of previous pixels. From this point on, our algorithm works on these maps as they now hold the characterization of all patches. Therefore, when an action is said to be taking place on patches, actually, the action takes place on these maps until the process of remapping which is a part of post-processing (Section 3.2.3). The pseudocode of this initialization phase is given in Algorithm 1.

The next step in our algorithm is seed identification. Here, the aim is to find connected components CC of patches on the image that constitute a homogeneous region using the constructed maps, that will also be used as a basis for

Algorithm 1 Initialization

Input: Image I, trained CNN, overlap distance d
Output: posterior map $\mathcal{P}_{map}$, label map L_{map}

procedure INITIALIZATION
    $row \leftarrow 0$
    $column \leftarrow 0$
    while $row < height_I$ **do**
        while $col < width_I$ **do**
            $p_i \leftarrow I[row : row + height_P - 1, col : col + width_P - 1]$
            $\{\pi_k(p_i)\}^6{}_{k=1} \leftarrow$ CNNForwardPass(p_i)
            $l(p_i) = \underset{k}{\text{argmax}}(\pi_k(p_i))$
            for $k = 1$ to 6 **do**
                $\mathcal{P}_{map}(p_i, k) \leftarrow \pi_k(p_i)$
            end for
            $L_{map}(p_i) \leftarrow l(p_i)$
            $col \leftarrow col + d$
        end while
        $row \leftarrow row + d$
    end while
end procedure

region growing. A p_i is selected as a seed patch candidate if it satisfies the following property: The posterior corresponding to its label must be greater than a threshold p_{thr}. The connected components are found on the patch candidates with the same label and the largest N connected components are determined as initial seeds. This step is detailed in Algorithm 2. The initialization and seed identification steps are illustrated in Figure 3.10.

Algorithm 2 Seed Identification

Input: posterior map $\mathcal{P}_{map}$, label map L_{map}, number of initial seeds N, posterior threshold p_{thr}

Output: initial seeds $\mathcal{S}$

procedure SEEDIDENTIFICATION
  $cand_k \leftarrow \varnothing$
  for $k = 1$ to 6 **do**
    for $\forall p_i$ **do**
      if $L_{map}(p_i) == l_k$ and $\mathcal{P}_{map}(p_i, k) \geq p_{thr}$ **then**
        $cand_k \leftarrow cand_k \bigcup p_i$
      end if
    end for
    $cc_k \leftarrow$connectedComponents$(cand_k)$
    $CC \leftarrow CC \bigcup cc_k$
  end for
  $\mathcal{S} \leftarrow$selectLargest(CC, N)
end procedure

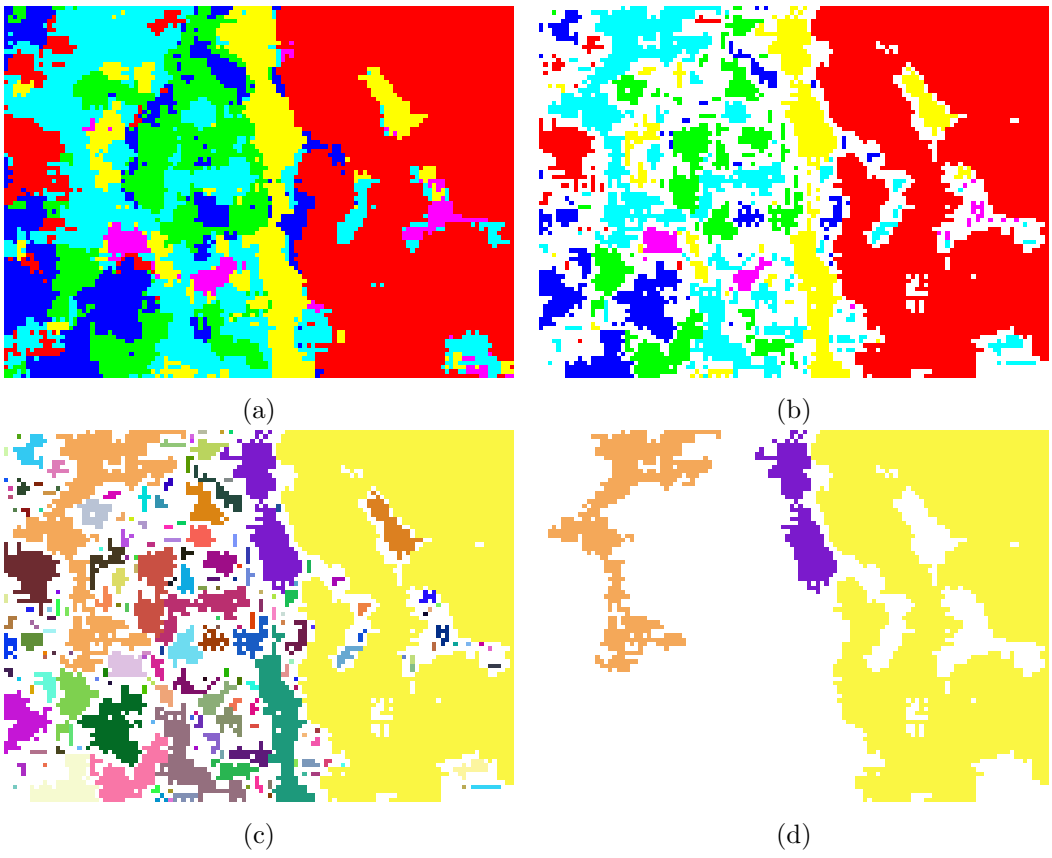


Figure 3.10: Illustration of initialization and seed identification steps, on a sample image: (a) Label map (L_{map}) of the overlapping patches. (b) Label map when patches with maximum posterior values smaller than p_{thr} are eliminated. The eliminated patches are shown with white. (c) Connected components (CC) of the remaining patches. Each connected component is shown with a different color. (d) The N largest connected components. Here, N is selected as 3.

3.2.2 Region Growing

The end product of seed identification step is a collection of seed regions r_j, examples of which are shown in Figure 3.10 (d). The next step is to grow these seed regions over unsegmented regions (which are shown in white in the same figure). To this end, at each iteration, until there is no unsegmented patch to grow upon, all neighboring patches of all grown regions are considered. From

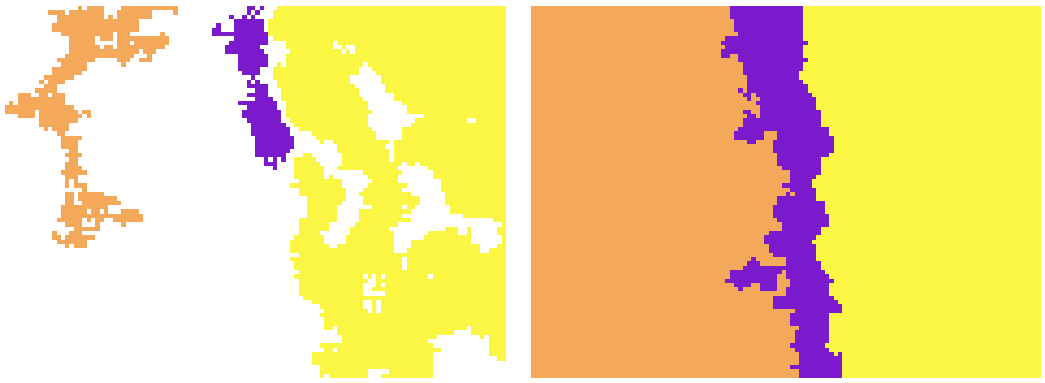
these patches, only one p_i is selected for merging into an r_j. This p_i has the highest similarity value $sim_{p_i,r_j} = \mathcal{P}_{map}(p_i, l_{r_j})$ where l_{r_j} is the class label of the patches that are initially in region r_j (before the growing process takes place). The pseudocode of the region growing step is given in Algorithm 3, and the operation is visualized in Figure 3.11.

Algorithm 3 Region Growing

Input: seeds $\mathcal{S}$, probability map $\mathcal{P}_{map}$, label map L_{map}

Output: grown regions $\mathcal{R}$

procedure RegionGrowing
  $\mathcal{R} \leftarrow \mathcal{S}$
  for $r_j \in \mathcal{R}$ **do**
    $l_{r_j} \leftarrow L_{map}(\text{ any } p_i | \{p_i \in r_j\})$
  end for
  while $\exists p_i | \{p_i \notin \mathcal{R}\}$ **do**
    $max_{sim} \leftarrow 0$
    for $r_j \in \mathcal{R}$ **do**
      for $p_i | \{p_i$ is adjacent to $\exists p_m \in r_j, p_i \notin \mathcal{R}\}$ **do**
        $sim_{p_i,r_j} = \mathcal{P}_{map}(p_i, l_{r_j})$
        if $sim_{p_i,r_j} > max_{sim}$ **then**
          $max_{sim} \leftarrow sim_{p_i,r_j}$
          $p_i^* \leftarrow p_i$
          $r_j^* \leftarrow r_j$
        end if
      end for
    end for
    $r_j^* \leftarrow r_j^* \bigcup p_i^*$
  end while
end procedure



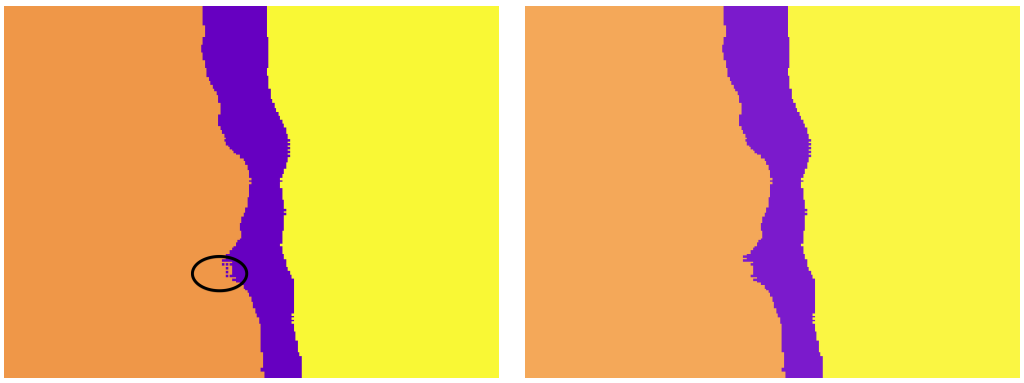
(a) (b)

Figure 3.11: Illustration of the region growing step, on a sample image. (a) Determined seeds after the seed identification step. (b) Resulting segmentation after the region growing step.

3.2.3 Post-processing

As mentioned, previous steps were carried out on the maps ($\mathcal{P}_{map}$ and L_{map}) that characterize the overlapping patches. However, due to the overlapping, there is no one-to-one correspondence between this map's entries and image pixels. Therefore, the first operation that this step accomplishes is the remapping of the segmented map entries back to the image pixels. For this, we use a voting algorithm, where every p_i submits a (single) vote in favor of its segmentation id to all of its pixels. The result is a voting map $\mathcal{V}$, that holds votes for all segmentation ids, for all pixels. When every patch is considered and every pixel's vote is counted, each pixel is assigned to the segmented region with the highest vote. The result of this remapping procedure (that is applied to the grown regions illustrated in Figure 3.11 (b)) is illustrated in Figure 3.12 (a).

The remapping procedure can produce extra small regions as byproducts (such as the ones indicated by a circle in Figure 3.12 (a)), especially in the region borders. The next procedure therefore in the post-processing step is the clean-up of these small regions. For this, all (now updated by voting) r_j are considered. If the area of a r_j is smaller than the area threshold a_{thr}, it is merged with its largest neighbor. Figure 3.12 (b) illustrates the result of this procedure. Notice that there are no small extra regions anymore. This clean-up procedure in post-processing



(a) (b)

Figure 3.12: Illustration of the post-processing step on a sample image: (a) Resulting regions after the remapping operation. The circled area includes small regions that are byproducts of the remapping procedure. (b) Resulting segmentation after the extra small regions are eliminated by clean-up.

step finalizes the Tier 2 operation, and outputs an unsupervised segmentation.

Chapter 4

Experiments

We conduct our experiments on the microscopic histopathological images of colon tissues. We quantatively evaluate our method, comparing its results with those of the others. In this chapter, first, datasets that are used both in the design of our methodology and its evaluation are discussed (Section 4.1). Then, our evaluation technique and metrics are presented (Section 4.2). Next, the methods that we use in comparisons are briefly explained (Section 4.3). Before presenting the results, a discussion is made about the parameters and their selection ranges (Section 4.4). Finally, the results are presented (Section 4.5), followed by a discussion on these experimental results (Section 4.6).

4.1 Dataset

In our experiments, we use two independent datasets: *AuxiliarySet*, which contains patch size annotated homogeneous images, and *SegmentationSet*, which contains larger unannotated heterogeneous images. In this work, we address the problem of unsupervised segmentation of the heterogeneous images of *SegmentationSet*. For this aim, we use the supplementary *AuxiliarySet* in the supervised training/fine-tuning of our convolutional neural network (CNN) model.

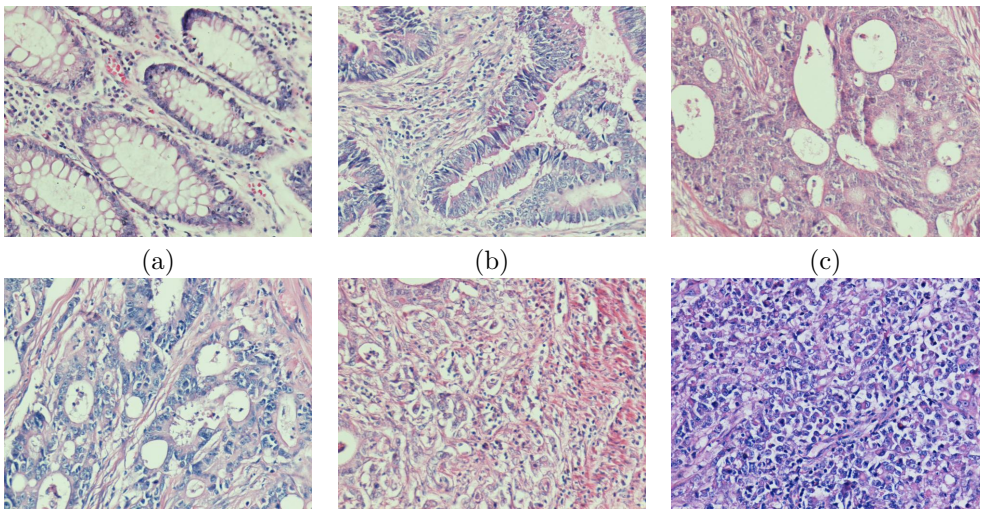


Figure 4.1: Each image in *AuxiliarySet* roughly belongs to one of the following categories: (a) Normal, (b) grade 1 cancer, (c) at the boundary between grade 1 and grade 2 cancer, (d) grade 2 cancer, (e) high grade cancer, and (f) medullary cancer.

Both datasets contain microscopic colon tissue images stained with hematoxylin-and-eosin. The biopsies were taken from the Pathology Department Archives of Hacettepe Medical School and their images were acquired by a Nikon Coolscope Digital Microscope. *AuxiliarySet* contains 900 RGB images acquired by using 20× microscope objective lens. Each image roughly correspond to one of the six categories; containing normal or abnormal formations. An example for each category is given in Figure 4.1. The resolution of these images is 480×640. For the usage in Tier 1, these images were cropped (in random positions) to the resolution of 227×227, the default input dimension of AlexNet.

The *SegmentationSet* contains 200 RGB images of resolution 1920×2560, acquired by using 5× microscope objective lens. Fifty out of these 200 images were reserved as a training set to estimate method parameters. The remaining 150 were reserved as a test set to evaluate the method's performance. These heterogeneous images contain both normal and adenocarcinomatous (cancerous) regions of different grades. An example is provided in Figure 4.2. Note that the same training and test sets were used in our previous studies [23].

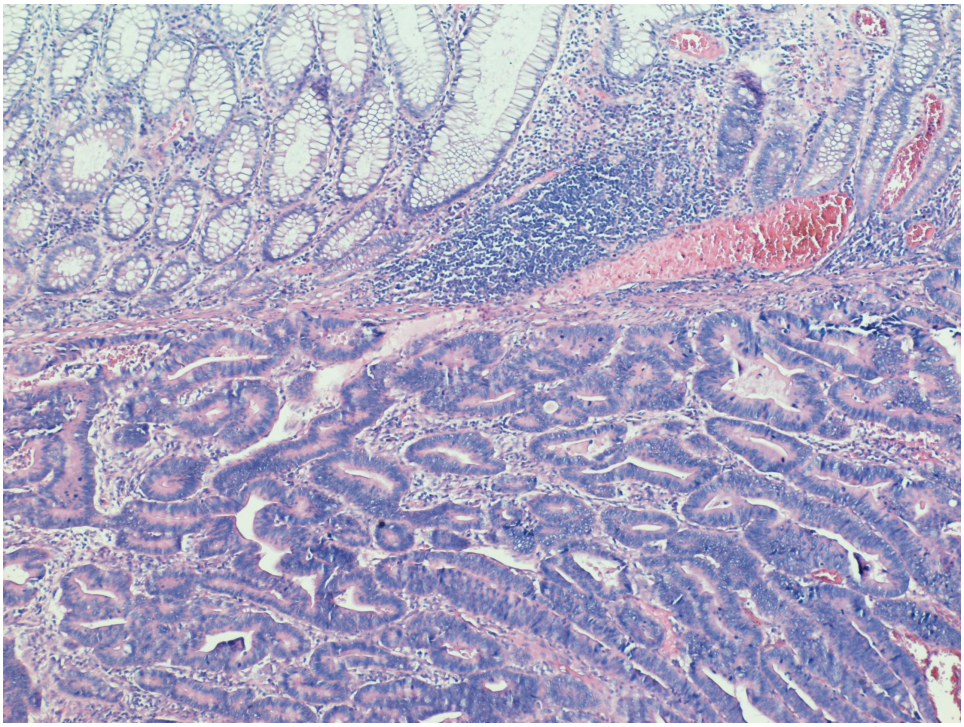


Figure 4.2: A sample image from *SegmentationSet*.

The last important point in discussing two sets is their *independence*. That is, images in *AuxiliarySet* are not cropped from the larger images of *SegmentationSet*. This independence is particularly important; because if they were not, the embedded supervised information from Tier 1 would create a positive bias in improving the unsupervised segmentation in Tier 2. Especially, this would result in an (erroneously) improved evaluation, as the segmentation images in that case would not be unseen.

4.2 Evaluation

We quantitatively evaluate the segmentation performance of the proposed *deepSeg* method using the gold standard versions of the images from *SegmentationSet*, provided by our medical collaborator. The gold standard versions are delineated by considering the colon adenocarcinoma, which accounts for 90-95% of all colorectal cancers. This cancer type originates from glandular epithelial cells and

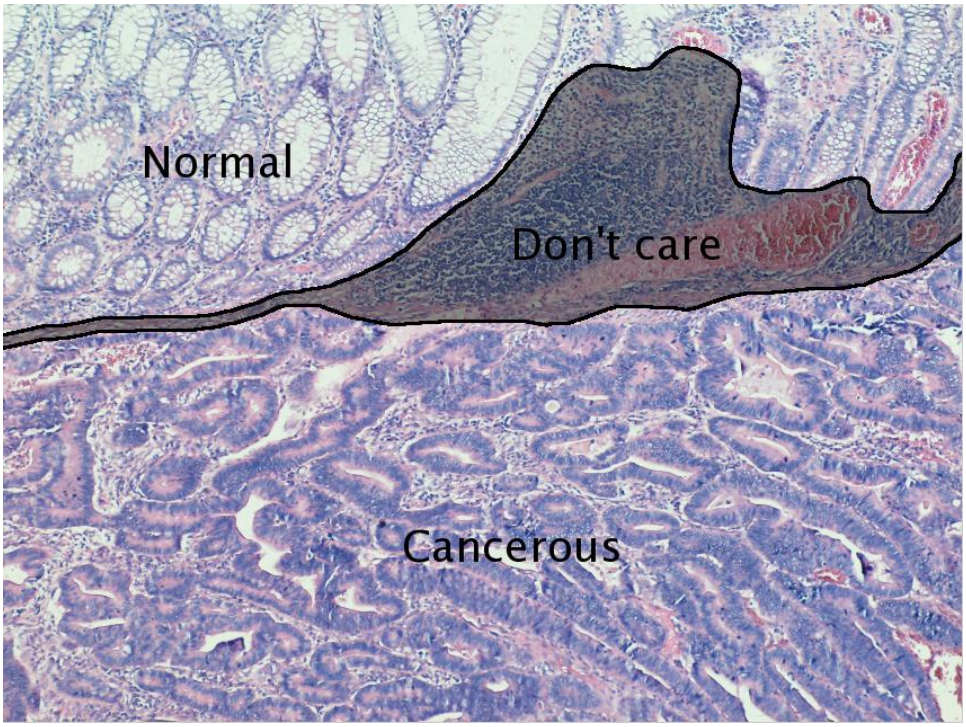


Figure 4.3: Gold standard version of the image shown in Figure 4.2.

causes deformations in colon glands. Non-glandular regions are not important in the context of colon adenocarcinoma diagnosis. Therefore, these regions could be included into either the normal part or the cancerous part. An example is provided in Figure 4.3. In this image, the upper part contains normal regions and the lower one consists of adenocarcinomotous regions. In the middle, which is the shaded part, there exists a region whose characteristics are not important in the context of colon adenocarcinoma diagnosis. Since this thesis focuses on colon adenocarcinoma, such regions will be considered as *don't care* regions in the methods' evaluation.

Our segmentation method, as well as the others used in comparisons, are all unsupervised, so they do not label their segmented regions as normal or cancerous. For this reason, for the evaluation, each segmented region r_j is attached to the label of its most overlapping region g_k in the gold standard. Then, the overlapping pixels of r_j and g_k are considered as *true positives* (TP), if g_k corresponds

to a cancerous region. If it corresponds to a normal region, these pixels are considered as *true negatives* (TN). Following the same understanding, non-overlapping pixels of r_j are considered as *false positives* (FP) if g_k corresponds to a cancerous region, and they are considered as *false negatives* (FN) if g_k corresponds to a normal region. The pixels of the *don't care* region in the gold standard are not considered in this evaluation. Using these four numbers, four evaluation metrics are calculated: accuracy, sensitivity, specificity, and f-score; whose definitions are given in the following equations.

$$\text{accuracy} = (TP + TN)/(TP + TN + FP + FN) \tag{4.1}$$

$$\text{sensitivity} = TP/(TP + FN) \tag{4.2}$$

$$\text{specificity} = TN/(TN + FP) \tag{4.3}$$

$$\text{precision} = TP/(TP + FP) \tag{4.4}$$

$$\text{recall} = TP/(TP + FN) \tag{4.5}$$

$$\text{f-score} = 2 \times (\text{precision} \times \text{recall})/(\text{precision} + \text{recall}) \tag{4.6}$$

4.3 Comparisons

In order to compare our results with those of the others, we use the following five methods: *MLSeg* [23], *graphRLM* [22], *objectSEG* [21], graph-based segmentation (*GBS*) [32], and J value segmentation (*JSEG*) [33]. The first three are the ones that were previously developed in our research group for the specific purpose of tissue segmentation. These algorithms all decompose an image into a set of circular objects. They then define handcrafted textural features on these objects to quantify their spatial distribution and use these handcrafted features in the segmentation. The comparison of the proposed *deepSeg* with these algorithms enables to understand the effects of using deep learning based features instead of the handcrafted ones.

The *GBS* and *JSEG* algorithms are known as effective general-purpose segmentation algorithms, so they are not specially designed for segmenting tissue images. These algorithms also employ handcrafted features for segmentation. The comparison with these two methods enables to understand the importance of using domain specific knowledge (which we introduce in Tier 1, when we fine-tune our model using tissue images), as well as observe the effects of using deep learning based features. The details of these five algorithms are given below.

- Multilevel segmentation (*MLSeg*) algorithm: A tissue image is decomposed into a set of circular tissue objects, which approximately represent tissue components. Then the *object cooccurence features* are extracted on these objects by calculating the frequency of cooccurrence of two object types with different distances. Consequently, a weighted graph is constructed between the objects where an edge in between two objects corresponds to the similarity between the features of these objects. Finally, a multilevel graph partitioning algorithm is used on this graph to obtain the segmentation [23].

- Graph run-length matrix (*graphRLM*) algorithm: Similarly in this algorithm, an image is decomposed into tissue objects. Then, a graph is constructed on these objects and its edges are colored according to their endpoints. Using these colored edges, a *run length matrix* is generated on this graph. The features extracted from this run length matrix are used in a region growing algorithm to obtain the segmentation [22].

- Object oriented segmentation (*objectSEG*) algorithm: This method also defines circular tissue objects and defines two uniformity measures on these objects. These are the object size uniformity and object spatial distribution uniformity, which quantify how the objects are distributed in size and space, respectively. Then those features are used in a region growing algorithm to segment a tissue image [21].

- Graph-based segmentation (*GBS*) algorithm: This method constructs a graph on an image, where the nodes are pixels and the edges are the dissimilarity values between the pixels (based on their properties)– with the

overall aim of obtaining low dissimilarity values among the elements in a component, and high dissimilarity values in different components. Based on this predicate, this method uses a greedy algorithm for segmentation [32].

- J value segmentation (*JSEG*) algorithm: This two-stage method accomplishes color quantization in the first phase to create a label map of an image. Then, according to a defined spatial criterion J on good segmentation, this method minimizes this criterion in the second phase. For that, it first reconstructs the image based on local J values as pixel values. Then, it performs a region growing algorithm on this reconstructed image by the steps of seed determination, seed growing, and region merge to obtain the segmentation [33].

4.4 Parameter Selection

The first tier of our method contains the hyper-parameters of the used convolutional neural network (CNN) model. Its second tier has the parameters related with the seed-controlled region growing algorithm. These parameters and their selected values are explained in the following subsections.

4.4.1 Parameters of Tier 1

We use transfer learning to fine-tune the AlexNet model in Tier 1. If we use a new model, we would need to make decisions for all aspects related to the CNN. This includes, but is not limited to the number and types of layers, the kernel sizes, pads and strides for convolutional and pooling layers, the number of outputs that convolutional and fully connected layers produce, and nonlinearity, pooling operation types– essentially every parameter discussed in Section 2.1. In the context of learning algorithms, these parameters are called *hyper-parameters*. More formally, a hyper-parameter set is a set $\mathcal{H}$ of parameters, which must be chosen (not derived) in order to obtain the actual learning algorithm [34].

In our case however, as we decide to use transfer learning (mainly due to the limited nature of our dataset) on the exact AlexNet model, only model's size related hyper-parameter that must be altered is the last fully connected (decision) layer's number of outputs, which is updated as six to correspond to the number of classes represented in *AuxiliarySet*, so there is no need for a selection. We use the AlexNet model as it is (except its last layer) because of the following reasons. Due to the similarities of our image scales, we do not want to leave out, replace or add any layers, with the logic that a model that has appropriate depth, width and layer properties to fairly learn an image dataset should not fail to learn a different dataset that consists of images which are similar in resolution. Therefore, instead of substantially modifying AlexNet, we crop our *AuxiliarySet* images at random positions to meet the ImageNet input size. This usage also enables us to transfer the maximum possible amount of information, from the unmodified layers of AlexNet. The rationale for using AlexNet architecture rather than a variation is validated when the evaluation results are presented for multiple models (Section 4.6).

However in fine-tuning, we had to consider the differences of both datasets– ImageNet and our histopathological tissue image dataset, and selected the learning parameters to successfully learn our dataset. We use five learning parameters in defining the training operation. These are the base learning rate $base_\eta$, learning rate multiplier γ, learning rate update stepsize S, total number of iterations num_{iter}, and momentum α. The selected values of these parameters are: $base_\eta = 0.001$, $\gamma = 0.8$, $S = 2000$, $num_{iter} = 15,000$, and $\alpha = 0.9$. These parameters are mainly selected by trial-and-error. When a parameter set was selected, the training is allowed to continue for a limited time. At that time, the changes in the error function output and validation images' accuracy are observed. If the change is satisfactory, meaning that the classification error is decreasing in a desirable rate, the iterations are allowed to continue. These parameters are taken from the best such operation. Here, note that the validation images are also selected among *AuxiliarySet*, so *SegmentationSet* images that are used for evaluation are not used at all. Similarly, the rationale of using transfer learning rather than learning from scratch is validated when the evaluation results of both

approaches are presented (Section 4.6).

4.4.2 Parameters of Tier 2

Tier 2 has the following external model parameters: The first one is the probability threshold p_{thr}, which is used to determine the homogeneous seeds in the seed identification step (Section 3.2.1). When a patch has a smaller maximum posterior value than this threshold, it is not considered in a seed region. The second parameter is the area threshold a_{thr}, which is used to eliminate small segmented regions that are close to region boundaries, in the clean-up procedure of post-processing (Section 3.2.3). When a segmented region's area is smaller than this threshold, it is merged to its largest adjacent region.

To select the values of these parameters, we obtain the segmentation results for all combinations of $p_{thr} = \{0.75, 0.90, 0.95\}$ and $a_{thr} = \{1000, 5000, 10000, 50000\}$ and select the combination that yields the best F-score metric on the training images of the *SegmentationSet*. Note that we use the same approach to select the parameters of the comparison algorithms. The details of this selection can be found in [22, 23].

In addition to these external model parameters, there are a couple of internal choices. These are the size of the sliding window (determined by $height_P$ and $width_P$) and the overlap distance d– both used to divide an image into overlapping patches of $m \times n$ in the initialization step (Section 3.2.1). We determine the size of the sliding window same as the input size of AlexNet, which is 227×227. We select the overlap distance as 20 in our experiments, based on trial-and-error on the training set.

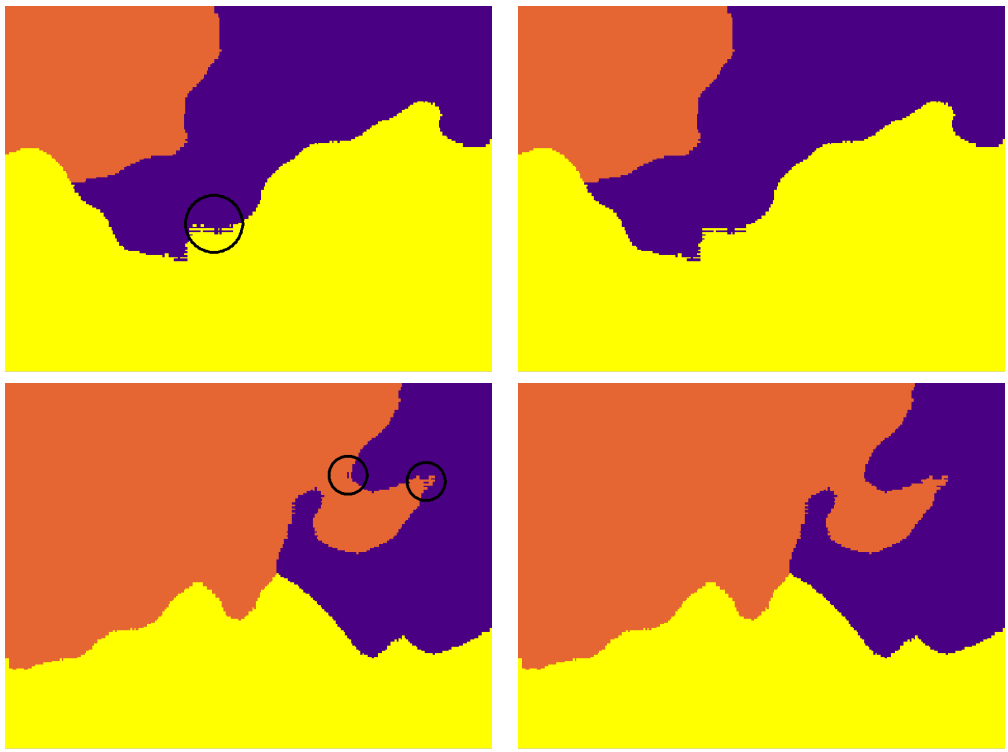
4.5 Results

Before presenting the results, there is an important point to consider. Both our proposed method *deepSeg* and the comparison method *MLSeg* segment an image into N regions. This is a fixed number that is same for all images. The number of segmented regions in the other segmentation algorithms is controlled by their model parameters (such as merge threshold) and this number is dynamically selected for each image separately. The difficulty in this approach is the difficulty of selecting good parameter values for multiple images, which result in good evaluation metrics while obtaining a satisfactory number of the segmented regions. Typically, the parameter combinations that give high accuracies lead to *oversegmentation*– a large number of the segmented regions that loses the ability to convey meaningful visual information. Thus, in our previous studies, we enforced the parameter selection in these algorithms to only consider the combinations that do not give segmented regions more than an upper-bound M. In the previous studies [21, 22], M is selected as 5 and 10. For these values, the comparison algorithms usually result in 3-4 and 5-6 segmented regions, respectively (see Table 4.2). To compare these methods fairly with *deepSeg* and *MLSeg*, we set the values of N as 3 and 5. Note that the selection of $N = 3$ is also consistent with the characteristics of the images in the *SegmentationSet*, as they originally contain two or three regions to be segmented in this context.

Table 4.1: Quantitative results on the training set. The best results (based on F-score) for $N = 3$ and $N = 5$ are indicated in bold.

			p_{thr}		
N	a_{thr}	metric	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	**92.94 ± 9.17**	92.22 ± 9.95	91.51 ± 10.92
		sensitivity	**93.27 ± 15.08**	90.73 ± 19.92	89.23 ± 23.17
		specificity	**91.46 ± 19.38**	93.39 ± 14.20	93.00 ± 14.18
		f-score	**92.65 ± 10.85**	90.66 ± 16.70	89.01 ± 19.97
	5000	*accuracy*	92.94 ± 9.17	92.22 ± 9.95	91.51 ± 10.92
		sensitivity	93.27 ± 15.08	90.73 ± 19.92	89.23 ± 23.17
		specificity	91.46 ± 19.38	93.39 ± 14.20	93.00 ± 14.18
		f-score	92.65 ± 10.85	90.66 ± 16.70	89.01 ± 19.97
	10000	*accuracy*	92.94 ± 9.17	92.22 ± 9.95	91.51 ± 10.92
		sensitivity	93.27 ± 15.08	90.73 ± 19.92	89.23 ± 23.17
		specificity	91.46 ± 19.38	93.39 ± 14.20	93.00 ± 14.18
		f-score	92.65 ± 10.85	90.66 ± 16.70	89.01 ± 19.97
	50000	*accuracy*	92.94 ± 9.17	92.22 ± 9.95	91.49 ± 10.93
		sensitivity	93.27 ± 15.08	90.73 ± 19.92	89.23 ± 23.17
		specificity	91.46 ± 19.38	93.39 ± 14.20	92.96 ± 14.23
		f-score	92.65 ± 10.85	90.66 ± 16.70	89.00 ± 19.97
5	*1000*	*accuracy*	94.06 ± 8.12	94.19 ± 7.86	**94.39 ± 7.20**
		sensitivity	94.15 ± 12.65	94.46 ± 11.67	**95.07 ± 10.12**
		specificity	94.04 ± 13.98	93.95 ± 13.99	**93.68 ± 13.96**
		f-score	93.94 ± 8.87	94.16 ± 8.18	**94.45 ± 7.04**
	5000	*accuracy*	94.06 ± 8.12	94.19 ± 7.86	94.39 ± 7.20
		sensitivity	94.15 ± 12.65	94.46 ± 11.67	95.07 ± 10.12
		specificity	94.04 ± 13.98	93.95 ± 13.99	93.68 ± 13.96
		f-score	93.94 ± 8.87	94.16 ± 8.18	94.45 ± 7.04
	10000	*accuracy*	94.06 ± 8.12	94.19 ± 7.86	94.39 ± 7.20
		sensitivity	94.15 ± 12.65	94.46 ± 11.67	95.07 ± 10.12
		specificity	94.04 ± 13.98	93.95 ± 13.99	93.68 ± 13.96
		f-score	93.94 ± 8.87	94.16 ± 8.18	94.45 ± 7.04
	50000	*accuracy*	94.02 ± 8.18	94.16 ± 7.91	94.37 ± 7.22
		sensitivity	94.09 ± 12.83	94.44 ± 11.75	95.07 ± 10.12
		specificity	94.01 ± 14.04	93.91 ± 14.05	93.63 ± 14.02
		f-score	93.88 ± 8.98	94.12 ± 8.25	94.44 ± 7.06

The quantitative results of the proposed *deepSeg* method on the training images of the *SegmentationSet* are given in Table 4.1. These results are the averages and the standard deviations obtained on the training set, considering different values of p_{thr} and a_{thr}. From this table, one can observe that a_{thr} does not affect the quantitative results. Indeed, it only changes the segmentation for a couple of images, for which the remapping procedure produces small extra regions close to the boundaries (see Figure 4.4). Since these extra regions are small and sometimes in the *don't care* regions, they do not alter the quantitative results significantly– only slightly alter the visual results. For example, only a small change after the decimal point is observed for $N = 5$ and $a_{thr} = 50000$. Thus, we select $a_{thr} = 1000$ for both $N = 3$ and $N = 5$. Table 4.1 also shows that the probability threshold affects the results, especially for N=3. Considering these evaluation results, which are obtained on the training set, we select $p_{thr} = 0.75$ for $N = 3$ and $p_{thr} = 0.95$ for $N = 5$. The parameter analysis for p_{thr} is given in Section 4.6.



(a) (b)

Figure 4.4: Visual effects of the area threshold a_{thr}. (a) Before merging the regions that are smaller than a_{thr}. The circles include small regions that are subject to merging. (b) After merging the small regions.

With these selected parameter values, the quantitative test set results of the proposed *deepSeg* method are given in the first rows of Tables 4.2 (a) and 4.2 (b), comparatively with other algorithms. Here note that, although N is a fixed number for *deepSeg*, the method yields four segmented regions for one image (out of 150 test images) when $N = 5$. In this case, the average number of regions is 4.99 and its standard deviation is 0.08. This is due to the post-processing step, as the small segmented regions are merged to their largest neighbors.

From these comparison results (Tables 4.2 (a) and 4.2 (b)), it is revealed that the proposed *deepSeg* method improves the segmentation results of all other comparison methods. Recall that *MLSeg*, *GraphRLM* and *ObjectSEG* use high-level object based representations, where they define handcrafted features. So the improvement over these methods is mainly attributed to the effectiveness of using

Table 4.2: Comparative quantitative results for the test set: (a) The segmented region number $N = 3$ for the *deepSeg* and *MLSeg* algorithms and the upper bound $M = 5$ for the other comparison algorithms. (b) The segmented region number $N = 5$ for the *deepSeg* and *MLSeg* algorithms and the upper bound $M = 10$ for the other comparison algorithms.

	Method	**Accuracy**	**Sensitivity**	**Specificity**	**F-score**	**Region no**
$N = 3$	*deepSeg*	94.9 ± 7.0	96.2 ± 9.1	92.0 ± 16.8	95.1 ± 7.1	3.0 ± 0.0
	MLSeg	92.9 ± 7.6	94.2 ± 12.4	90.3 ± 15.8	92.9 ± 10.7	3.0 ± 0.0
$M = 5$	*GraphRLM*	84.8 ± 14.4	85.8 ± 26.2	76.2 ± 35.7	81.5 ± 26.6	2.8 ± 1.1
	ObjectSEG	86.9 ± 11.5	90.4 ± 21.8	77.2 ± 28.5	78.6 ± 27.9	4.1 ± 1.3
	GBS	73.4 ± 8.9	64.5 ± 33.8	77.2 ± 30.3	58.5 ± 37.7	3.7 ± 1.2
	JSEG	69.4 ± 12.1	62.7 ± 45.5	62.2 ± 39.8	46.2 ± 38.4	2.9 ± 1.3

(a)

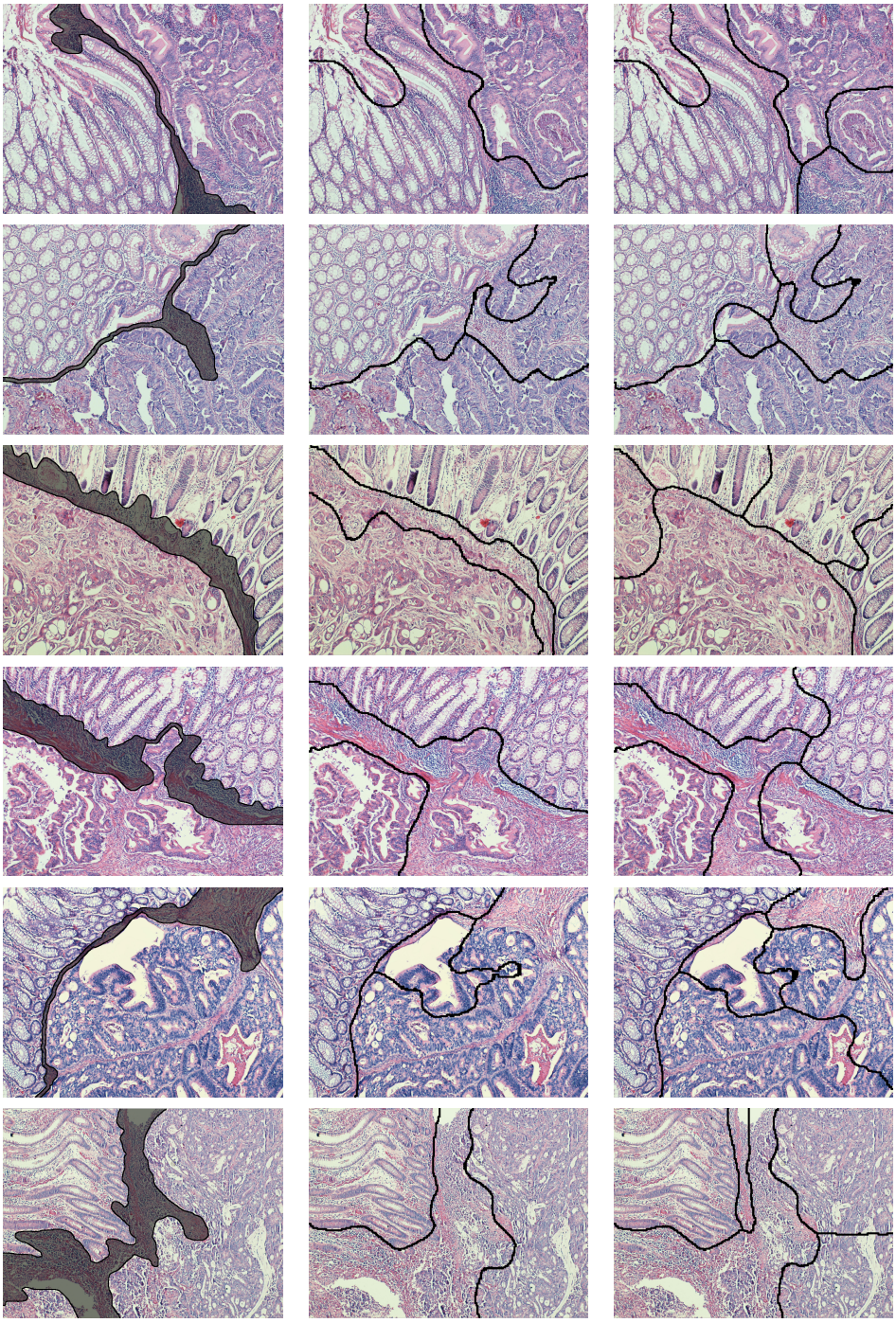
	Method	**Accuracy**	**Sensitivity**	**Specificity**	**F-score**	**Region no**
$N = 5$	*deepSeg*	96.2 ± 5.2	96.7 ± 8.4	94.7 ± 8.6	96.2 ± 6.0	5.0 ± 0.1
	MLSeg	94.9 ± 5.5	95.8 ± 7.1	92.9 ± 9.3	95.2 ± 5.6	5.0 ± 0.0
$M = 10$	*GraphRLM*	91.5 ± 9.8	92.3 ± 15.6	87.3 ± 24.2	91.3 ± 13.8	5.6 ± 1.6
	ObjectSEG	89.9 ± 11.4	89.3 ± 23.5	85.8 ± 22.7	87.8 ± 19.1	5.8 ± 2.0
	GBS	74.0 ± 9.5	64.7 ± 31.9	74.5 ± 28.7	61.9 ± 32.0	5.0 ± 1.7
	JSEG	89.6 ± 7.2	89.1 ± 16.3	87.1 ± 18.2	87.6 ± 12.5	7.9 ± 2.7

(b)

deep-learning based features. This fact is also applicable to the last two comparison methods of *GBS* and *JSEG*, as they also use handcrafted features. However, again recall that these two algorithms are not domain specific. Therefore, the improvement over these algorithms also indicates the effectiveness of using the domain specific knowledge, which is accomplished by transfer learning in our method.

After presenting the quantative results of the proposed *deepSeg* method on the test set, for a visual understanding, Figure 4.5 is provided. This figure includes visual segmentation results for *deepSeg*, for both $N = 3$ and $N = 5$. The parameters a_{thr} and p_{thr} are selected in accordance to the N values, as discussed previously. In this figure, there are particularly interesting segmentation results. For example, the third segmentation result for $N = 3$, divides the image into three regions in a very similar fashion to the gold standard. So in this case, it is able to distinguish between normal, cancerous, and *don't care* regions. For $N = 5$,

the algorithm diversified the normal region, where visually different looking subregions are separated. As another example, the fifth segmentation manages to separate normal and cancerous regions, for $N = 3$. The white region at the center is also separated from the rest of the image, as it is visually different. When the number of the segmented regions are increased to 5, again the method obtains a very accurate segmentation, but this time, also the non-glandular subregion on the top-left is separated from the normal region. These examples, alongside with the others indicate that the proposed *deepSeg* algorithm is able to obtain very accurate segmentations in terms of obtaining the two segmented regions (excluding the *don't care* region) for $N = 3$. For $N = 5$, the method is able to further distinguish visually different regions, on top of obtaining the segmentations for normal and cancerous regions.



(a) (b) (c)

Figure 4.5: Visual results of the *deepSeg* method, on six sample images: (a) Gold standards of the images. Gray-shaded parts indicate *don't care* regions and their pixels are not considered in quantitative evaluation. (b) Segmented regions for $N = 3$. (c) Segmented regions for $N = 5$.

4.6 Discussion

Next, we investigate the effectiveness of using the AlexNet model and transferring the weights learned on the ImageNet dataset for the fine-tuning. For this purpose, we take the AlexNet architecture and train it from scratch (no information is transferred from the ImageNet dataset and the initial weights of the model are randomly selected). As we use 15 000 iterations in our *deepSeg* method, we firstly use the same number of iterations for the scratch model. Then, we increase that number to 30 000 to see its effects on the performance of the scratch model. Table 4.3 (a) specifies the quantitative test set results of the *deepSeg* method, where the CNN model's initial weights are transferred from the AlexNet model trained on the ImageNet dataset, and the weights are fine-tuned using our *AuxiliarySet*. Then, Table 4.3 (b) and 4.3 (c) provide the quantitative test set results for the model trained from scratch on our *AuxiliarySet* for 15 000 and 30 000 iterations, respectively. Note that in this table, we provide results for different values of p_{thr} when a_{thr} is fixed to 1000. As the results of the other values of a_{thr} are almost the same, we exclude them from this table for the sake of simplicity.

The results reveal that the fine-tuned model significantly outperforms the models with the same architecture, but trained from scratch. The fine-tuned model is trained for 15 000 iterations, and it can be observed that the results are significantly better for all parameter values, compared to the model that is trained for 15 000 iterations from scratch. As there is no transfer of information in this model, further iterations might be needed for the weights to converge. Thus, to ensure a fair learning process, we trained this model for another 15 000 iterations to obtain a model trained for 30 000 iterations. This further training however, does not create a significant difference with the model that is trained from scratch for 15 000 iterations. For some parameters the F-score results are better, for others worse. Comparing the fine-tuned model and the model that is trained for 30 000 iterations from scratch, the fine-tuned model obtains superior results in approximately half the training time, therefore the fine-tuned model is also more efficient. This difference in performance is attributed to the transfer of information, learned from the extensive natural images dataset. As mentioned,

Table 4.3: Quantitative results obtained on the test set when the initial weights of the model are: (a) Transferred from the ImageNet dataset and fine-tuned for 15 000 iterations, (b) randomized and trained for 15 000 iterations, and (c) randomized and trained for 30 000 iterations.

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	94.94 ± 6.96	94.51 ± 7.44	94.69 ± 7.55
		sensitivity	96.18 ± 9.07	95.56 ± 9.75	95.79 ± 10.61
		specificity	92.02 ± 16.75	92.25 ± 15.50	92.34 ± 15.36
		f-score	95.14 ± 7.14	94.74 ± 7.53	94.79 ± 8.37
5	*1000*	*accuracy*	96.32 ± 5.27	96.43 ± 5.19	96.22 ± 5.23
		sensitivity	97.42 ± 5.93	97.28 ± 6.34	96.74 ± 8.41
		specificity	94.13 ± 11.31	94.48 ± 10.98	94.69 ± 8.60
		f-score	96.52 ± 4.72	96.62 ± 4.80	96.20 ± 5.99

(a)

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	89.48 ± 9.91	89.09 ± 10.08	88.37 ± 10.19
		sensitivity	92.00 ± 16.00	91.62 ± 15.22	90.10 ± 17.38
		specificity	81.58 ± 28.60	82.02 ± 28.28	82.46 ± 28.06
		f-score	89.68 ± 12.42	89.39 ± 12.27	88.30 ± 14.18
5	*1000*	*accuracy*	93.06 ± 7.23	92.82 ± 7.31	92.72 ± 7.41
		sensitivity	94.46 ± 10.22	94.46 ± 8.77	93.66 ± 11.89
		specificity	89.53 ± 17.22	88.94 ± 18.75	89.64 ± 18.17
		f-score	93.30 ± 8.13	93.27 ± 7.00	92.67 ± 10.33

(b)

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	89.46 ± 9.92	89.11 ± 10.23	88.45 ± 10.24
		sensitivity	92.63 ± 15.35	91.47 ± 16.73	90.06 ± 17.65
		specificity	80.71 ± 28.86	81.71 ± 28.54	82.20 ± 28.32
		f-score	89.80 ± 12.22	89.12 ± 14.10	88.33 ± 14.29
5	*1000*	*accuracy*	93.21 ± 7.18	93.15 ± 7.22	92.74 ± 7.07
		sensitivity	94.58 ± 10.39	94.19 ± 9.40	93.58 ± 11.92
		specificity	89.26 ± 17.65	90.17 ± 16.88	89.55 ± 17.77
		f-score	93.43 ± 8.01	93.47 ± 7.29	92.69 ± 10.09

(c)

CNNs are shown to learn low-level image feature extractors in the earlier layers. This information is also shown in effect as our fine-tuning operation aimed to preserve the information in the earlier layers.

After investigating the effects of fine-tuning, we investigate the effects of using different architectures, rather than the exact AlexNet model. For this aim, two new architecture variations are created. Variation 1 is obtained by removing third and fourth convolutional-nonlinearity layer pairs. Variation 2 is obtained by adding a convolutional-nonlinearity layer pair at the end of the fifth convolutional-nonlinearity layer pair. These variations are intended to reflect the effects of using shallower and deeper architectures. As the architecture of AlexNet is altered, these models are trained from scratch, first for 15 000 iterations. Then, these models are trained for another 15 000 iterations to reach 30 000 training iterations in total– the understanding here is the same with the previous scratch model. Table 4.4 (a) indicates the test set results for the 15 000 iterations version of Variation 1, and Table 4.4 (b) indicates the test set results for the 30 000 iterations version. Likewise, Table 4.5 (a) indicates the test set results for the 15 000 iterations version of the Variation 2, and Table 4.5 (b) indicates the test set results for its 30 000 iterations version.

Table 4.4: Quantitative results on the test set by using Variation 1 in which the third and fourth convolutional-nonlinearity layer pairs are removed, when the model is trained for: (a) 15 000 iterations, (b) 30 000 iterations.

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	88.38 ± 11.03	88.10 ± 11.27	87.81 ± 11.31
		sensitivity	88.70 ± 18.88	88.02 ± 20.09	87.73 ± 21.07
		specificity	84.07 ± 26.61	82.60 ± 29.56	82.67 ± 29.06
		f-score	87.98 ± 14.47	87.31 ± 16.34	86.81 ± 16.99
5	*1000*	*accuracy*	93.29 ± 7.91	92.51 ± 8.13	92.22 ± 8.12
		sensitivity	91.89 ± 15.45	90.89 ± 16.12	90.92 ± 15.88
		specificity	92.20 ± 17.01	91.15 ± 18.53	90.89 ± 18.33
		f-score	92.45 ± 12.34	91.70 ± 12.62	91.51 ± 12.48

(a)

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	88.25 ± 10.99	87.96 ± 11.22	88.30 ± 10.90
		sensitivity	88.76 ± 18.63	87.28 ± 20.72	88.13 ± 20.18
		specificity	83.12 ± 27.43	82.86 ± 29.29	83.03 ± 28.73
		f-score	87.94 ± 14.32	86.95 ± 16.56	87.40 ± 16.29
5	*1000*	*accuracy*	92.95 ± 8.04	92.00 ± 8.82	92.18 ± 8.26
		sensitivity	92.05 ± 14.21	90.14 ± 16.64	91.36 ± 14.96
		specificity	91.34 ± 18.66	90.78 ± 19.92	90.41 ± 18.85
		f-score	92.43 ± 11.43	91.24 ± 12.82	91.71 ± 11.86

(b)

Table 4.5: Quantitative results on the test set by using Variation 2 in which a convolutional-nonlinearity layer pair is added after the fifth convolutional-nonlinearity layer pair, when the model is trained for: (a) 15 000 iterations, (b) 30 000 iterations.

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	90.85 ± 10.06	91.34 ± 9.01	90.92 ± 9.36
		sensitivity	93.30 ± 15.75	92.63 ± 16.93	92.34 ± 17.49
		specificity	85.29 ± 23.10	86.18 ± 21.03	85.66 ± 21.72
		f-score	90.75 ± 13.65	90.63 ± 14.83	90.21 ± 15.14
5	*1000*	*accuracy*	93.88 ± 6.86	94.14 ± 6.14	94.16 ± 6.19
		sensitivity	94.83 ± 11.31	95.44 ± 8.82	95.11 ± 10.56
		specificity	90.80 ± 16.17	90.88 ± 15.33	91.40 ± 14.28
		f-score	93.81 ± 9.28	94.33 ± 6.70	94.07 ± 8.75

(a)

			p_{thr}		
N	a_{thr}	**metric**	*0.75*	*0.90*	*0.95*
3	*1000*	*accuracy*	90.59 ± 10.08	91.03 ± 9.25	90.74 ± 9.29
		sensitivity	92.14 ± 16.59	92.16 ± 17.24	92.39 ± 17.12
		specificity	86.47 ± 22.36	86.62 ± 20.39	85.14 ± 22.70
		f-score	90.32 ± 13.79	90.31 ± 14.94	90.18 ± 14.80
5	*1000*	*accuracy*	93.56 ± 7.17	93.97 ± 6.33	94.01 ± 6.42
		sensitivity	94.24 ± 11.84	95.09 ± 10.82	94.87 ± 10.84
		specificity	91.00 ± 15.95	90.77 ± 15.62	91.21 ± 15.07
		f-score	93.46 ± 9.56	93.92 ± 8.82	93.92 ± 8.95

(b)

When the quantitative results for both variations (Tables 4.4 and 4.5) are compared with the results of the fine-tuned model (Table 4.3 (a)), again the fine-tuned model significantly outperforms both model variations, regardless of their number of training iterations. This is again mainly attributed to the transfer of

information from AlexNet, where our selection of the architecture enables us to transfer the maximum possible amount of information, as we transfer information for every layer except the last decision layer. Here, note that again increasing the number of iterations do not create a significant change, for both variations; and the deeper Variation 2 performs better, compared to the shallower Variation 1.

Finally, we also analyze the effects of the probability threshold p_{thr} to the segmentation performance of the proposed *deepSeg* method. For this, we obtain the segmentation results of the test set for the extended values of $p_{thr} = \{0, 0.25, 0.50, 0.55, 0.60, \ldots, 0.90, 0.95, 0.99, 1.00\}$. The area threshold in this experimentation is fixed to $a_{thr} = 1000$. Figure 4.6 shows the obtained F-score metric as a function of p_{thr} for $N = 3$ and $N = 5$.

This figure indicates that a high F-score could be obtained for smaller p_{thr} values, until a peak at a high value, and poor F-scores are obtained for its higher values. Recall that this parameter is used to eliminate low confidence patches from being selected as seeds. This result is an indication that our CNN model makes confident patch classifications, as the results are stable until very high threshold values. When the threshold is selected excessively high, it is natural for the results to drop, as the model cannot produce enough seed regions. This is due to the impracticality for neural networks to produce classifications with full confidence.

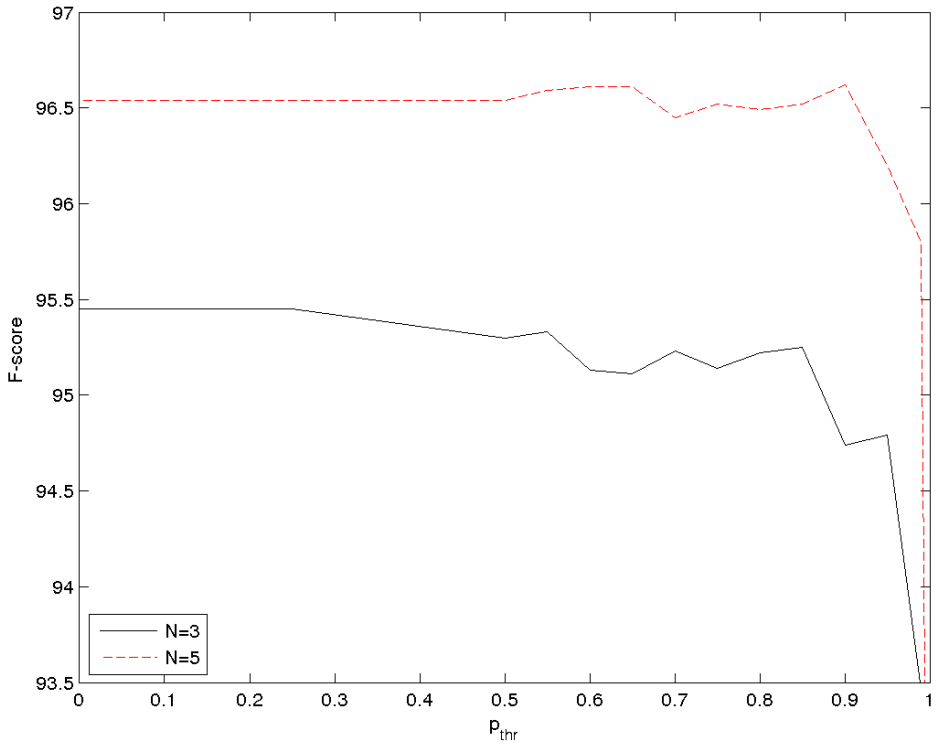


Figure 4.6: Test set F-score metrics as a function of the probability threshold p_{thr}. Area threshold is fixed to $a_{thr} = 1000$.

Chapter 5

Conclusion

The main contributions of the *deepSeg* method that we propose in this thesis are twofold. First, it uses deep learning to characterize or produce features of histopathological tissue images, despite the majority of previous studies, where they use task specific handcrafted features for characterization. Second, it transfers the knowledge from a non-medical domain of natural images to the histopathological image characterization for segmentation. With these two properties, it is one of the first studies that attempt to merge deep learning and histopathological image segmentation in this manner. Furthermore, in Tier 2 of our method, instead of simply connecting same labeled patches (as it is done by the previous studies in this domain), we develop a seed-controlled region growing algorithm that works on the labeled patches. Overall, we devise a method that manages to outperform other methods on the same task, and thus marks an improvement in the domain of medical image segmentation, to hopefully have a positive influence on the efforts against colon adenocarcinoma.

Our experiments on histopathological tissue images yield that the proposed *deepSeg* method outperforms other methods, and this effectiveness is attributed to using deep learning based features and transferring the information learned on natural images to the domain of medical image segmentation. Additionally, the comparisons with the AlexNet model trained from scratch, as well as two

varying architectures; where one is selected as shallower and the other one as deeper to enable a more thorough analysis, reveal that the approach of fine-tuning the pre-trained AlexNet model significantly outperforms other approaches for convolutional neural network (CNN) training.

To sum up, our proposed *deepSeg* method presents an automatic, task independent two-tier methodology that would potentially enable it to work on any image segmentation task. This is primarily due to the fact that none of the operations that we use are limited to the task in hand; Tier 1 uses a CNN that is shown to be effective in a range of image processing tasks and Tier 2 uses automated region growing. To make the method suitable for any other image segmentation task, transfer learning can be used to transfer the knowledge gained from the natural images to that specific image domain. However, we do not make any performance investigations in the context of this study. Therefore, a possible extension would be to investigate this method in different types of image datasets, where it is also possible to consider different datasets for transfer learning (with varying CNN models) to show that the results that this method produce are satisfactory. If not, the method can be modified to better suit the confines of new datasets. Overall, we believe that we produced a generalizable method, which can be modified to be used in an array of image segmentation tasks, both for other medical images and images of very different domains, such as segmenting regions on satellite image data.

Bibliography

[1] I. Nagtegaal and J. Van Krieken, "The role of pathologists in the quality control of diagnosis and treatment of rectal cancerian overview," *European Journal of Cancer*, vol. 38, no. 7, pp. 964–972, 2002.

[2] P. Quirke, "Limitations of existing systems of staging for rectal cancer: the forgotten margin," in *Rectal Cancer Surgery*, pp. 63–81, Springer, 1997.

[3] I. Ellis, "I cannot really envisage pathology without computational pathology in the future."

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[5] J. J. Hopfield and D. W. Tank, "Computing with neural circuits- a model," *Science*, vol. 233, no. 4764, pp. 625–633, 1986.

[6] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[7] Y. Bengio, Y. LeCun, *et al.*, "Scaling learning algorithms towards ai," *Large-scale Kernel Machines*, vol. 34, no. 5, pp. 1–41, 2007.

[8] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and Cooperation in Neural Nets*, pp. 267–285, Springer, 1982.

[9] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[10] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[11] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724, 2014.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, pp. 3320–3328, 2014.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[16] M. Mete, X. Xu, C.-Y. Fan, and G. Shafirstein, "Automatic delineation of malignancy in histopathological head and neck slides," *BMC Bioinformatics*, vol. 8, no. 7, p. S17, 2007.

[17] Y. Wang, D. Crookes, O. S. Eldin, S. Wang, P. Hamilton, and J. Diamond, "Assisted diagnosis of cervical intraepithelial neoplasia (cin)," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 1, pp. 112–121, 2009.

[18] D. Romo, E. Romero, and F. González, "Learning regions of interest from low level maps in virtual microscopy," *Diagnostic Pathology*, vol. 6, no. 1, p. S22, 2011.

[19] K. Mosaliganti, F. Janoos, O. Irfanoglu, R. Ridgway, R. Machiraju, K. Huang, J. Saltz, G. Leone, and M. Ostrowski, "Tensor classification of n-point correlation function features for histology tissue segmentation," *Medical Image Analysis*, vol. 13, no. 1, pp. 156–166, 2009.

[20] N. Signolle, M. Revenu, B. Plancoulaine, and P. Herlin, "Wavelet-based multiscale texture segmentation: Application to stromal compartment characterization on virtual slides," *Signal Processing*, vol. 90, no. 8, pp. 2412–2422, 2010.

[21] A. B. Tosun, M. Kandemir, C. Sokmensuer, and C. Gunduz-Demir, "Object-oriented texture analysis for the unsupervised segmentation of biopsy images for cancer detection," *Pattern Recognition*, vol. 42, no. 6, pp. 1104–1112, 2009.

[22] A. B. Tosun and C. Gunduz-Demir, "Graph run-length matrices for histopathological image segmentation," *IEEE Transactions on Medical Imaging*, vol. 30, no. 3, pp. 721–732, 2011.

[23] A. C. Simsek, A. B. Tosun, C. Aykanat, C. Sokmensuer, and C. Gunduz-Demir, "Multilevel segmentation of histopathological images using cooccurrence of tissue objects," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1681–1690, 2012.

[24] J. Xu, X. Luo, G. Wang, H. Gilmore, and A. Madabhushi, "A deep convolutional neural network for segmenting and classifying epithelial and stromal regions in histopathological images," *Neurocomputing*, vol. 191, pp. 214–223, 2016.

[25] J. Arevalo, A. Cruz-Roa, V. Arias, E. Romero, and F. A. González, "An unsupervised feature learning framework for basal cell carcinoma image analysis," *Artificial Intelligence in Medicine*, vol. 64, no. 2, pp. 131–145, 2015.

[26] F. Xing, Y. Xie, and L. Yang, "An automatic learning-based framework for robust nucleus segmentation," *IEEE Transactions on Medical Imaging*, vol. 35, no. 2, pp. 550–566, 2016.

[27] K. Sirinukunwattana, S. E. A. Raza, Y.-W. Tsang, D. R. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1196–1206, 2016.

[28] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[31] Y. Hamamoto, S. Uchimura, M. Watanabe, T. Yasuda, Y. Mitani, and S. Tomita, "A gabor filter-based method for recognizing handwritten numerals," *Pattern Recognition*, vol. 31, no. 4, pp. 395–400, 1998.

[32] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[33] Y. Deng and B. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800–810, 2001.

[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.